17583

NATIONAL LIBRARY
OTTAWA

CANADA

BIBLIOTHÈQUE NATIONALE
OTTAWA

NAME OF AUTHOR....Som Nath Kukreja.....

TITLE OF THESIS....Cellular Logic Arrays.....

.....

.....

UNIVERSITY....University of Alberta.....

DEGREE FOR WHICH THESIS WAS PRESENTED....M.Sc.....

YEAR THIS DEGREE GRANTED.....1973.....

Permission is hereby granted to THE NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed)...Som Nath Kukreja

PERMANENT ADDRESS:

.....4, Vijay Nagar.....

.....Jullundur City.....

.....Panjab, INDIA.....

DATED...Sept 20,....1973

NL-91 (10-68)

THE UNIVERSITY OF ALBERTA

CELLULAR LOGIC ARRAYS

by

© SOM NATH KUKREJA

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1973

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled "Cellular Logic Arrays," submitted by Som Nath Kukreja in partial fulfilment of the requirements for the degree of Master of Science.

.....
Supervisor

.....

.....

.....

Date..Sept. 6, 1973.....

ABSTRACT

Recent and anticipated advances in microelectronics presage a drastic reduction in the per-gate cost of circuitry when arranged in the form of a cellular array. The numerous engineering advantages of a cellular array have been described and justified by a number of authors. This thesis is mainly concerned with cellular logic arrays which can realize an arbitrary combinational switching function or an arbitrary synchronous sequential machine. The basic cell is a switching device with two inputs, two outputs and two control variables X and T. The cell structure is either purely combinational or with unit delay depending upon the control variable T. The control variable X sets up either a "crossing mode" or a "bending mode" in the cell. Thus a two dimensional cellular array without time delays has the connection capabilities of a crossbar switch. Any combinational switching function can be realized by appropriate choice of control variables while the inputs to the edges of the plane are fixed. Alternatively, the control variables can be fixed while the inputs to the edges of the cellular plane are varied from function to function. A cubic array is constructed from a set of identical cellular planes packed one upon the other such that the control variables applied to the first plane will penetrate to all other planes without time delay. It is shown that any K functions of the same variables can be synthesized on such a cubic array. By allowing the control variable T to delay some signals in the array, such a cubic array can be used

to realize any synchronous sequential machine with single or multiple inputs and/or feedback functions. Any defective cell in the array can be tested and isolated. The array can be stripped, divided, or interconnected.

A Rectangular sorting array is also described which is well adapted to realization by large-scale-integrated semiconductor technology. Each cell contains one flip-flop representing one digit of the binary coded word. Thus an array of dimensions MxN stores M words each of length N binary digits. It is shown that the file of M words can be sorted by pulsing the array approximately M/2 times. The largest or the smallest word can be read out from the file. With some additional circuitry lying external to the main array, the array can be used as an addressed, a pushdown, or a buffer memory. Finally, the array presents another isolated example in which feedback is used in the combinational circuitry. Some complex-cell sorting arrays are also described.

ACKNOWLEDGEMENT

I wish to express my gratitude to my supervisor, Dr. I. N. Chen, who has provided valuable assistance at every stage in the development of this thesis; also, I am indebted to the National Research Council for providing financial assistance under grant A7133. In addition, I would like to thank Dr. J. R. Sampson, Dr. K. A. Stromsmoe, and Prof. A. Wouk for their helpful suggestions concerning the final draft of this thesis.

TABLE OF CONTENTS

		Page
List of Tables		ix
List of Figures		x
CHAPTER		
1	INTRODUCTION TO CELLULAR LOGIC	1
	1.1 Batch Fabrication	2
	1.2 Combinational Cellular Structures	2
	A. Single-Rail Cascade	4
	B. Multiple-Rail Cascade	4
	C. Two Dimensional Cellular Arrays	6
	D. Three Dimensional Cellular Arrays	7
	1.3 Sequential Cellular Structures	7
2.	COMBINATIONAL CELLULAR STRUCTURES	11
	2.1 The Basic Cell	12
	2.2 Two Dimensional Cellular Arrays	14
	A. Calculation of Outputs	16
	B. Interconnection of Arrays	16
	C. Dividing of Arrays	18
	D. Stripping of Arrays	18
	E. Universal Logic Arrays	23
	(i) Fixed Inputs	23
	(ii) Fixed Control Variables	25
	(iii) Edge Fed Arrays	28
	2.3 Cubic Cellular Arrays	32
	A. Universal Cubic Array	32

Table of Contents, continued

CHAPTER		Page
	B. S-O-P and P-O-S synthesis	33
	2.4 Fault Testing	35
	2.5 Comparison	39
3.	SEQUENTIAL CELLULAR STRUCTURES	42
	3.1 Definite Machines	43
	3.2 Regular Machines	44
	(i) Single Feedback Realization	46
	(ii) Single Stage Realization	49
	3.3 Multiple Inputs, Outputs, and Feedback Variables	54
4.	SORTING ARRAYS	57
	4.1 The Basic Logical Array	57
	4.2 A Rectangular Sorting Array	64
	A. Addressed Memory	78
	B. Pushdown Memory	79
	C. Buffer Memory	79
	4.3 Fault Testing	79
	4.4 A Cubic Sorting Array	83
	4.5 Cell Complexity versus the Array Size . . .	88
5.	CONCLUSIONS	93
	REFERENCES	95

LIST OF TABLES

TABLE		Page
1.	Comparison of Cellular Arrays	41
2.	State Table of M_1	43
3.	State Table and Expanded Flow Table of M_2 using Single Feedback Realization Scheme . .	46
4.	State Table and Expanded Flow Table of M_2 Using Single Stage Realization Scheme	50
5.	Number Representation	68
6.	Interconnection Code and Output Cell Functions .	69
7.	Four Modes in the Rectangular Sorting Array . .	77
8.	Fault-detection in the Rectangular Sorting Array	81
9.	Fault-detection in the Rectangular Sorting Array	82

LIST OF FIGURES

FIGURE		Page
1.	Fabrication Methods for Conventional and Microcellular Integrated Circuits	3
2.	A Single-Rail Cascade	5
3.	A Two-Rail Cascade	5
4.	An Adder Array	8
5.	Realization of Sequential Behaviour	9
6.	Implementation of the Basic Cell	13
7.	The Basic Cell	15
8.	Cell operations under Different Controls	15
9.	Model of a Cell without Delay	15
10.	Outputs of a Cell under Different Inputs and Controls	17
11.	Information Flows in an Array	17
12.	Finding the Outputs of an Array	17
13.	Interconnecting of Arrays	19
14.	Dividing an Array	20
15.	Realization of Two Functions on a Cellular Plane	21
16.	Stripping an Array	22
17.	Universal Arrays for Two Variables C and D	24
18.	Universal Array for 3 variables B, C and D	24
19.	Universal Array for 4 variables A,B,C and D	24
20.	Reduction of Array Size	26
21.	Universal Arrays for 3 and 4 variables	26
22.	n Variables Universal Array Built from Universal Array for n-1 variables	27

List of Figures, continued

FIGURE		Page
23.	A Modified Cell Model, and Edge-Fed Universal Arrays for Two, Three and Four variables	29
24.	Other Version of Cell Model and Edge-Fed Universal Arrays	30
25.	A Universal Edge-Fed Array for Four Variables	31
26.	Cubic Array Realizing Sum-Of-Products	34
27.	Cubic Array Realizing Product-Of-Sums	34
28.	Detecting a Faulty Cell on a Path	38
29.	Isolating a Cell	38
30	Realization of M_1	45
31.	Single Feedback Realization of M_2	48
32.	Single Stage Realization of M_2	53
33.	Realization of a Synchronous Sequential Machine	56
34.	Cellular Sorting Array	58
35.	The Basic Cell	58
36.	Signal Flow in an Array	63
37.	Cellular Sorting Array	65
	(a) The Main Array	65
	(b) A Typical Cell and the Cell logic equations	66
38.	Cellular Sorting Array with Mask Register	73
	(a) The Main Array	73
	(b) A Typical Cell and the Cell logic equations	74
39.	Sorting Array and Addressed Memory	76
40.	Cubic Array for Sorting in the Space Domain	84
41.	Realization of a Set of Combinational Switching Functions	87

List of Figures, continued

FIGURE		Page
42.	Cellular Sorting Array	89
43.	Signal Flow in an Array	90
44.	Cellular Sorting Array	91

CHAPTER 1

INTRODUCTION TO CELLULAR LOGIC

This thesis is concerned with the analysis and design of some cellular logic arrays. The majority of them are of recent origin primarily from the impact of large-scale-integrated semiconductor technology in circuit design. In the past logical designers have constructed digital circuits with individual components such as resistors, capacitors and transistors. Such a circuit is termed a discrete circuit. The recent and anticipated advances in microelectronics promise a large quantity of interconnected components on a single wafer. This is commonly referred to as an integrated circuit. If the chip circuitry is arranged in the form of a cellular array, a number of advantages can be achieved. Some of them are circuit standardization, higher packing density, reduced wiring cost, easier fabrication, greater reliability and easier testability. The relation of cellular logic to batch fabrication is discussed in the next section. Sections 1.2 and 1.3 give a review of existing combinational and sequential cellular structures.

1.1 Batch Fabrication

Fig. 1 shows the ideal way of fabricating identical integrated circuits on a single silicon wafer by the process of photo-masking, etching and crystal growing [1]. These techniques will not be discussed here. However, we will discuss two different functional realization schemes. In fig. 1(a), the identical circuits are separated, attached in different packages and then interconnected in a particular fashion to obtain the required function. An example of this is the NAND or NOR cells realization of a combinational switching function.

Fig. 1(b) illustrates the batch fabrication technique for integrated circuits in which the functional realization is based on cellular logic. In this case, the duplicated circuits, i.e. cells, are uniformly interconnected according to some chosen regular fashion. Note that the duplicated circuits are not separated and, therefore, no packaging and re-assembling of the cells is required. The entire wafer is packaged as one module. The appropriate inputs to the edges of the array specializes the desired function. In some cases, each cell is provided with one or more external inputs. These are termed input parameters or cutpoint inputs. All these inputs can be taken to the terminals of the module through a deposit Conductor which forms an integral part of the module.

1.2 Combinational Cellular Structures

A cellular array consists of 1-, 2-, or 3-dimensional arrangement of cells interconnected in a regular fashion. Here we will review some of them briefly.

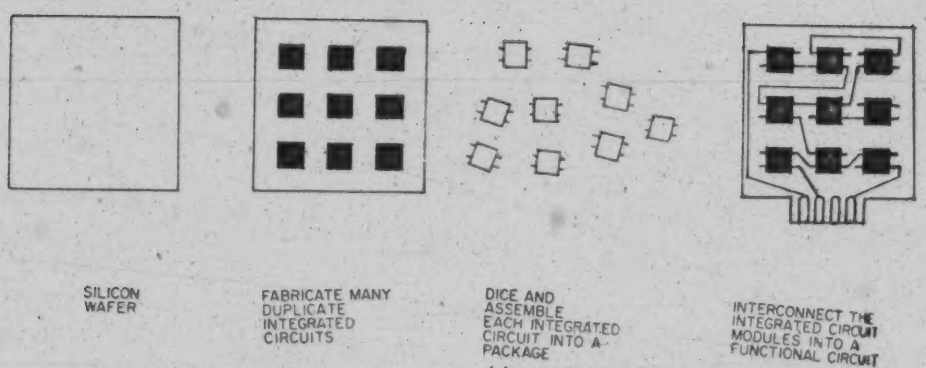

SILICON WAFER
FABRICATE MANY DUPLICATE INTEGRATED CIRCUITS
DICE AND ASSEMBLE EACH INTEGRATED CIRCUIT INTO A PACKAGE
(a)
INTERCONNECT THE INTEGRATED CIRCUIT MODULES INTO A FUNCTIONAL CIRCUIT
SILICON WAFER
FABRICATE ARRAY OF INTERCONNECTED GENERAL FUNCTION CELLS
(b)
SPECIALIZE THE ARRAY INTO A FUNCTIONAL CIRCUIT

FIG. 1. Fabrication methods for conventional and microcellular integrated circuits

A. Single-Rail Cascade

The simplest possible cellular structure can be obtained by cascading 2-input 1-output cells as shown in fig. 2. Each cell can produce one of three functions xy, x+y, x⊕y, where x,y are the inputs to the cell. The test for cascade realizability and the number of realizable functions have been discussed by Mukhopadhyay [3]. Fig. 2 shows the synthesis of the function.

$$F = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_4 + \bar{x}_2 x_3 \bar{x}_4$$

The theory of the single rail cascade has appeared in a number of papers [2, 4, 5, 6, 7, 8]. The various problems related to the single rail cascade are

1) To test for cascade realizability
2) To minimize the length of cascade
3) To determine the number of realizable functions

The first two problems have been solved by Sklansky [4], Levy et al. [5], Minnick [2] and Weiss [6]. Stone [7] and Sklansky et al. [8] have enumerated the functions realizable by a single-rail cascade.

B. Multiple-Rail Cascade

The single rail cascade is logically incomplete, i.e., there exist switching functions that can not be realized by such a cascade. It is possible to increase the logical capabilities by making the cascade multi-railed. Fig. 3 shows the realization of a combinational function $F = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_4 + \bar{x}_2 x_3 \bar{x}_4$ using a two-rail cascade.

The two-rail cascade is logically complete [3]. The various problems related to the two-rail cascade are

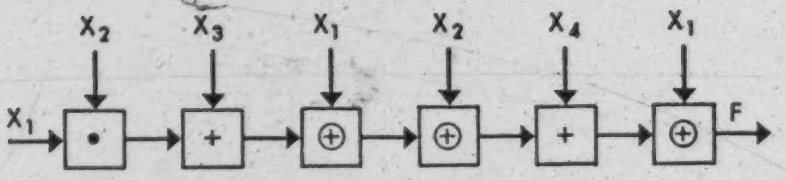
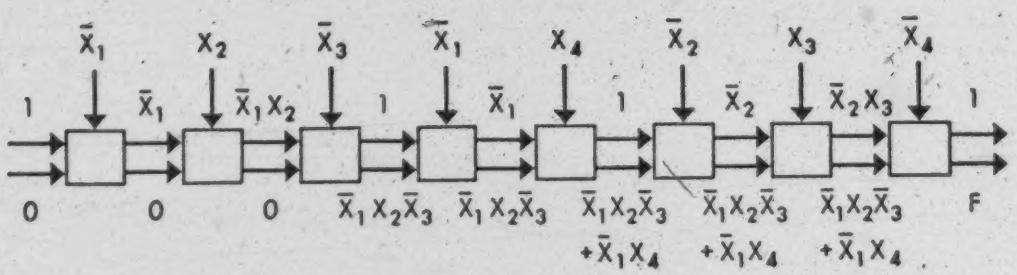


FIG. 2. A SINGLE-RAIL CASCADE



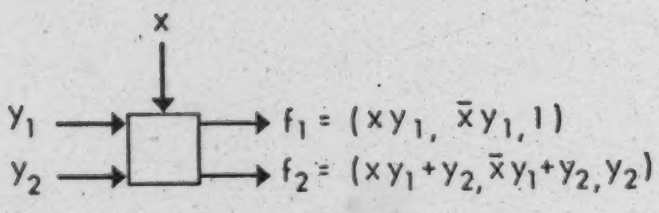


FIG. 3. A TWO-RAIL CASCADE

1) To determine an upper bound on the number of cells in a two-rail universal cascade.
2) To minimize the number of functions that must be produced in a typical cell.
3) To minimize the number of cells needed for the synthesis of a combinational switching function.

An approach to the first two problems is due to Short [9]. The third problem is still an open one. The theory of multi-rail cascade to synthesize a set of combinational switching functions has been studied by Elspas [10].

C. Two Dimensional Cellular Arrays

Two dimensional cellular arrays can be categorized on the basis of whether they are fixed cell-function or variable cell-function arrays. In the fixed cell-function arrays, the function produced by each cell is fixed. The cell parameters (i.e., cutpoint inputs) are used to modify the interconnection structure. In the variable cell-function arrays, the interconnection structure and the cell functions are determined by the input parameters. It should be noted that the fixed cell-function arrays can be interpreted as the variable cell function arrays.

In the fixed cell-function arrays, the combinational switching function is decomposed into a non-cellular tree of some fixed cell-function. The tree is then embeded into a rectangular cellular structure by the proper choice of input parameters. The input parameters define the interconnection structure which can be represented by a

graph. The properties of this graph and some NOR-NAND function arrays have been studied by Spandorfer and Murphy [11]. Majority or minority gate arrays also fall in the same category. These arrays have been studied by Short [12], Canaday [13, 14], Amarel, Cooke and Winder [15]. A simple fixed cell-function array known as the adder array is illustrated in fig. 4. The array produces all the min-terms in three variables x_1, x_2 and x_3. An arbitrary combinational switching function can be realized by collecting the appropriate min-terms in a cascade, each cell of which can produce one of the two functions $y+z$, y, where y,z are the inputs to the cell. The array was proposed by Kautz and Minnick in 1962 and was further studied by Minnick and Short [16].

The well known representatives of variable cell-function arrays are the Maitra Cascade [17], Minnick's cutpoint cellular arrays [2] and Cobweb arrays [18].

D. Three Dimensional Cellular Arrays

There has been no work done on three dimensional cellular arrays for the synthesis of combinational switching functions. In the next chapter we will develop a simple cubic array for the synthesis of a set of combinational switching functions.

1.3 Sequential Cellular Structures

Any multiple-output combinational array in conjunction with unit delays (exterior to the array) can be used for realization of an arbitrary sequential behaviour as shown in fig. 5. This is commonly referred to as single stage realization. Single feedback realization is

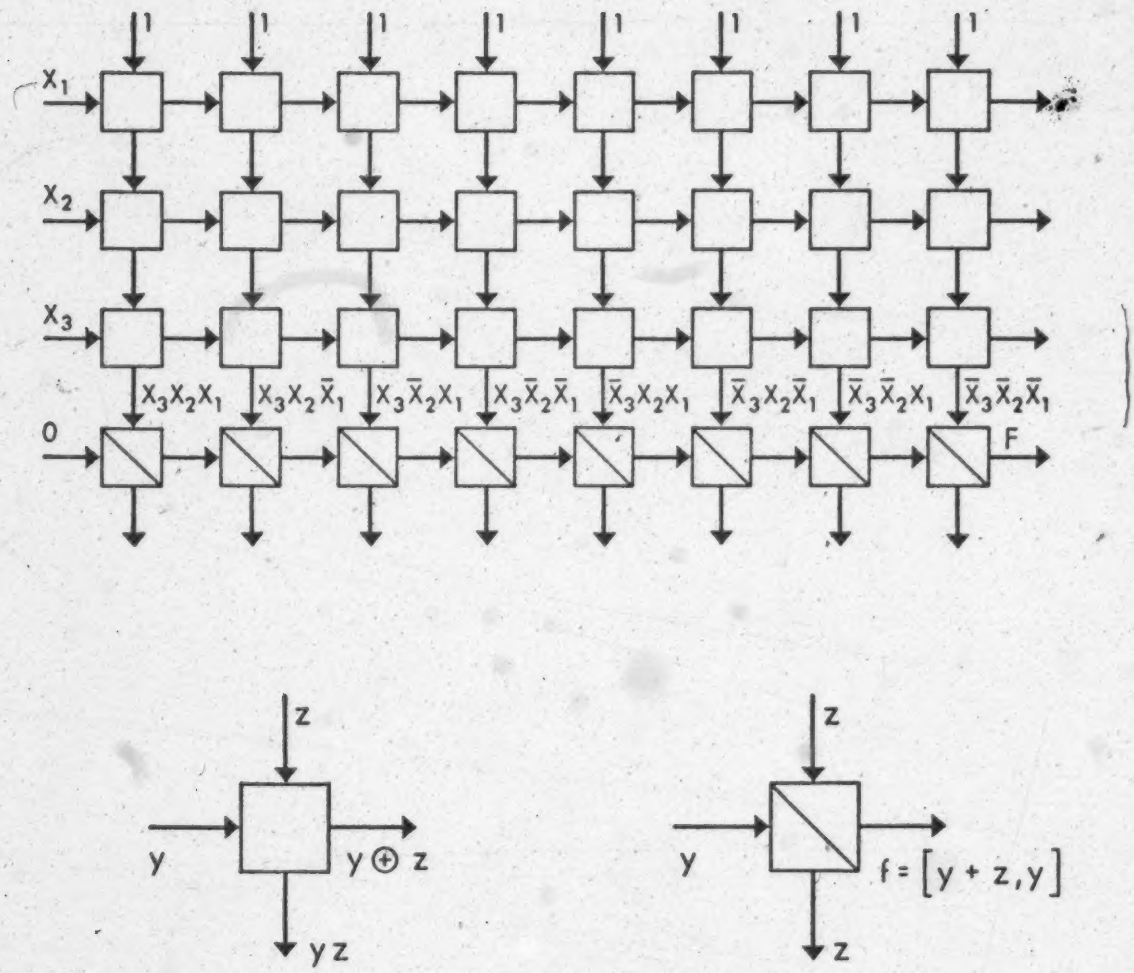


FIG. 4. AN ADDER ARRAY

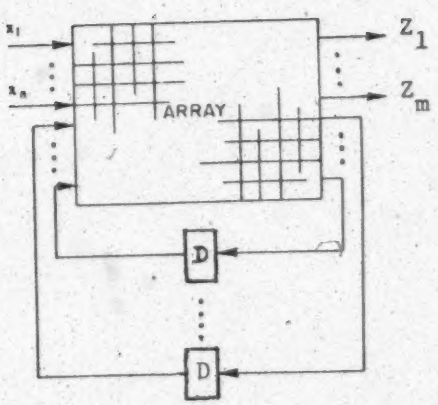


FIG. 5. REALIZATION OF SEQUENTIAL BEHAVIOR

possible in the sense described by Arnold et al. [19-21]. We will discuss these realizations in the third chapter. In all these arrays, state assigment is made without any restrictions. Minimization of the number of cells or the size of the array on the basis of proper state assignment has been developed by Ferrari and Grasseli [22]. However, no systematic method is available for minimization of the number of cells or total array size.

In the next chapter, a new cell model is proposed and its properties discussed. The basic cell is then used to construct universal logic arrays for realizing an arbitrary n variable combinational switching function. Some sequential cellular structures using the same basic cell are described in Chapter III. Chapter IV is self-contained and presents some sorting arrays suitable for implementation on LSI.

CHAPTER 2

COMBINATIONAL CELLULAR STRUCTURES

In this chapter we propose a new cell model and then construct some combinational cellular structures [30]. A universal logic array is defined, which consists of the basic cells interconnected in a regular fashion, that is capable of realizing every combinational switching function of n variables. Several methods for constructing universal logic arrays are described, where the functional specialization is achieved either by varying the input parameters of each cell or by changing the edge inputs to the array. Two modified versions of the basic cell yield interesting and efficient edge fed universal logic arrays. The problem of interconnecting, dividing, stripping and fault-detecting of these arrays is also discussed.

2.1 The Basic Cell

The basic cell is a switching device with two inputs, two outputs and two control variables. The inputs, y and Z, the outputs, f_1 and f_2, and the control variables, T and X, are all assumed to have logical values of 0 and 1. The functional capabilities of the cell can be best described by the following logical equations:

$$\left.\begin{aligned} f_1(t) &= X(t).Z(t-T) + \bar{X}(t).y(t-T) \\ f_2(t) &= X(t).y(t-T) + \bar{X}(t).Z(t-T) \end{aligned}\right\} \text{----------(1)}$$

One way of implementing such a cell is by the circuit shown in fig. 6(a) It consists of two unit delays and four identical S-switches. Each S-switch in turn consists of two AND gates and one OR gate as shown in fig. 6(b). First consider the output of the switch S_1 which is given by

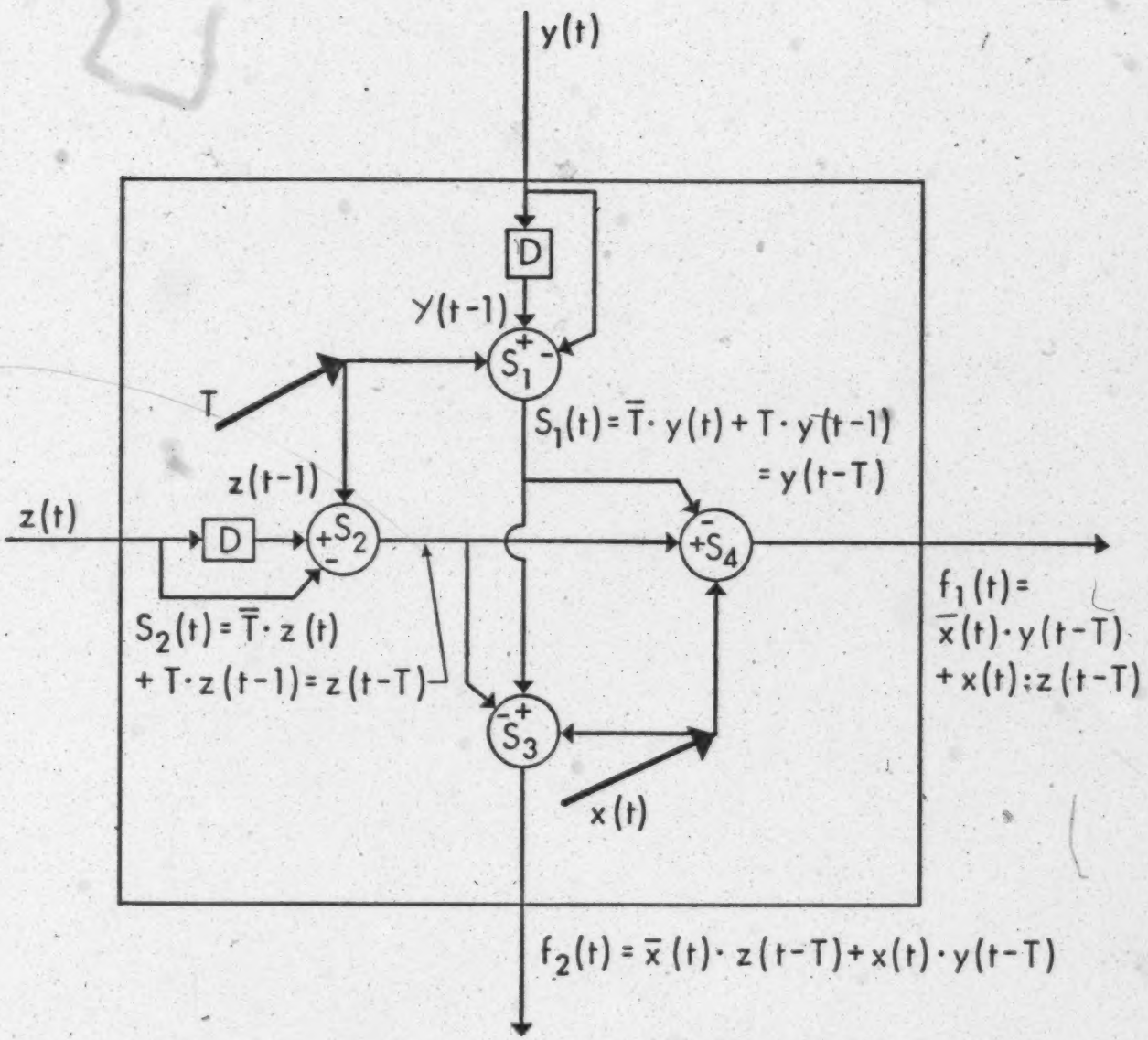
$$S_1(t) = \bar{T}.y(t) + T.y(t-1)$$

If $T = 0$, $S_1(t) = y(t)$, and if $T = 1$, $S_1(t) = y(t-1)$. Thus the output of the switch S_1 is either the current value or the unit-delayed value of the input y depending upon whether $T = 0$ or $T = 1$. A simple expression to represent this is

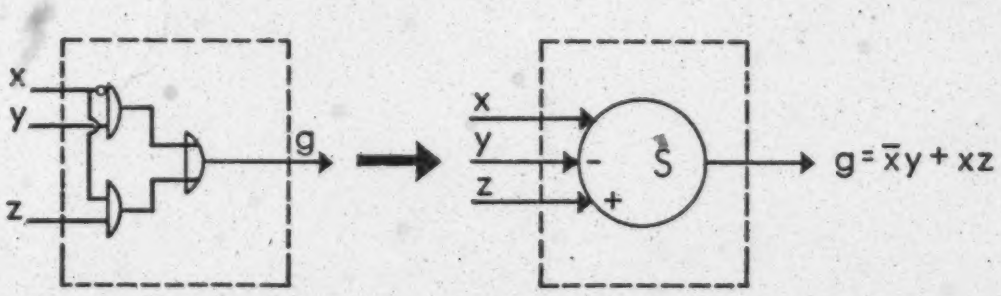
$$S_1(t) = y(t-T)$$

Similarly, $S_2(t) = Z(t-T)$. The outputs of the switches S_3 and S_4 can now be seen to be f_2 and f_1 respectively.

From now on we will neglect the internal structure of the cell and represent the cell model as shown in fig. 7. Under the control of T and x, the cell will perform the logical operations illustrated in



(a) INTERNAL STRUCTURE OF THE CELL



(b) THE S-SWITCH

FIG. 6. IMPLEMENTATION OF THE BASIC CELL

fig. 8, where [D] represents a unit delay.

Eq. (1) indicates that when T = 0, the outputs of the cell are functions of inputs and controls without time delay. The cell circuitry behaves in a purely combinational fashion and can be used to realize a combinational switching function. In this chapter, we will omit T and represent the cell model as shown in fig. 9. The logical operations of a cell under different combinations of inputs and controls are illustrated in fig. 10.

2.2 Two Dimensional Cellular Arrays

Here we describe some two dimensional cellular arrays for realizing an arbitrary combinational switching function. A cellular plane is composed of identical cells arranged in a rectangular or square grid. Each cell receives two inputs, one from its left neighbour and the other from the cell immediately above it. Likewise the two outputs of the cell are connected to the two cells immediately to the right and below. The external inputs to the cellular plane are inputs to the cells of the left-most column and the top row while the outputs from the plane are those from the cells of the right-most column and the bottom row. We will also assume that we have access to the control variables of the cellular plane. Furthermore, when dealing with a cellular plane we will omit inter-cell connections and represent the cellular plane as shown in fig. 11.

If the inputs and controls of the cellular plane are tied to 0 and 1, a particular configuration of information flows is obtained in the cellular plane. For example, one possible configuration in a

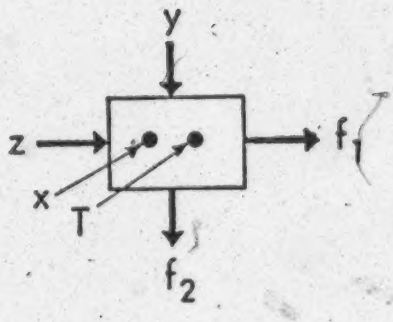


FIG. 7. THE BASIC CELL

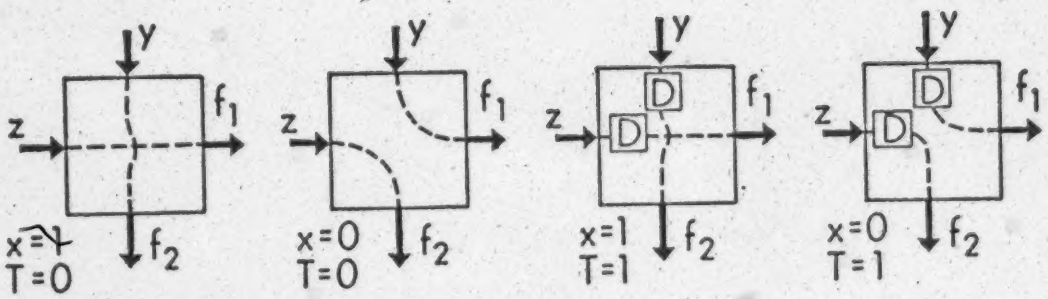


FIG. 8. CELL OPERATIONS UNDER DIFFERENT CONTROLS

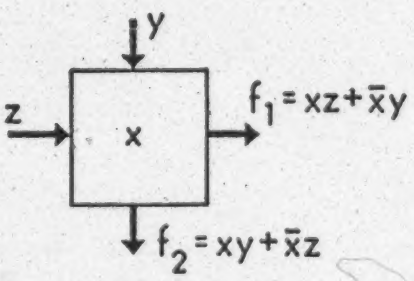


FIG. 9. MODEL OF A CELL WITHOUT DELAY

3 x 3 cellular plane is shown in fig. 11. Each cell is in a "crossing mode" or a "bending mode" [29] respectively. Thus, once all controls are fixed, the information flows in the network will not interfere. Fig. 11 also shows that the input 1 applied to the left of the top left corner cell will pass through the plane and exit from the right-most cell at the bottom edge. Similarly, the input 1 applied to the top centre cell will exit from the top right cell.

A. Calculation of Outputs

Any output will be 1 if and only if there is a logical 1 input at the edge of the plane and there is a path connecting this input lead to the output lead. Such a path is termed a 1-path. If the control variables are changed, a different configuration of 1-paths is induced in the cellular plane. Thus, knowledge of the inputs to the cellular plane and the control variables of all cells allows calculation of the output by searching through all possible 1-paths. The example of fig. 12 will be used for illustration. Since the upper cell has an external input of 1 and there is only one 1-path from it through the top row to the output lead for F_1 which is induced when A = 1, B = 1 and C = 0. We find that $F_1 = AB\bar{C}$. Similarly, $F_2 = \bar{A}BC$. For F_3, there are 6 possible 1-paths as indicated by dotted lines. Thus $F_3(A,B,C,D,E) = ABC\bar{D}E + A\bar{B}\bar{C}DE + A\bar{B}C\bar{D}\bar{E} + \bar{A}\bar{B}C\bar{D}E + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}D\bar{E}$

B. Interconnection of Arrays

It can be easily verified that if inputs to the cellular plane are complemented without changing the control variables, the function

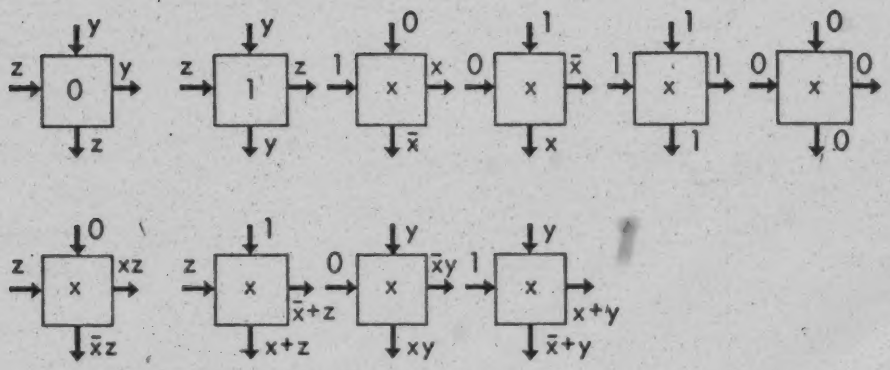


FIG. 10. OUTPUTS OF A CELL UNDER DIFFERENT INPUTS AND CONTROLS

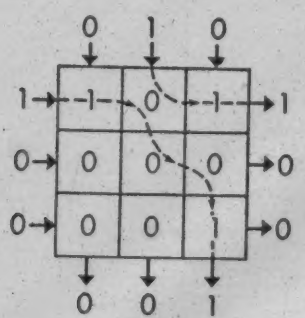


FIG. 11. INFORMATION FLOWS IN AN ARRAY

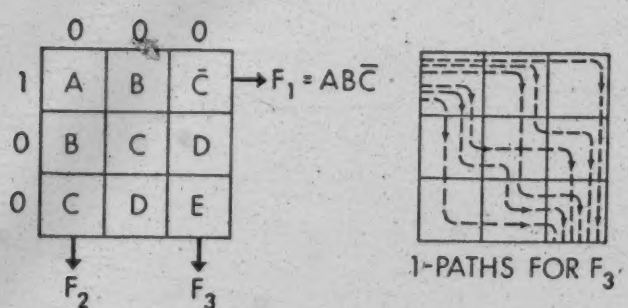


FIG. 12. FINDING THE OUTPUTS OF AN ARRAY

synthesized at the output lead is also complemented. The proof follows from the fact that any input combination of control variables induces one and only path connecting some input lead to the output lead. Interconnection of arrays can now be easily visualized. Assume that realizations of two separate functions F and G have been found to be M_1 and M_2. Fig. 13 shows how the functions $\bar{F}$, FG, $\bar{F}G$, $F + \bar{G}$ and $\bar{F} + \bar{G}$ can be realized. In fig. 13, a value inside a rectangle is the control output for all the cells within that area.

C. Dividing of Arrays

An array can be divided for realizing two or more independent functions. The method is to apply a logical 0 control to cells on a line parallel to the diagonal. These cells serve as a rebounding border and keep the information flows local to the regions. Fig. 14 illustrates the effect of dividing of an array. Fig. 15 displays realization of two functions F and G on a cellular plane.

D. Stripping of Arrays

Applying a logical 1 control to a cell will have the effect of stripping that cell from the array. The cell will be set to "crossing mode" and hence the inputs to the cell will pass through it without any deviation. A similar argument indicates that a column (or row) can be stripped by applying a logical 1 control to all cells on the column (or row) as shown in fig. 16. This property will later be used to isolate a defective cell.

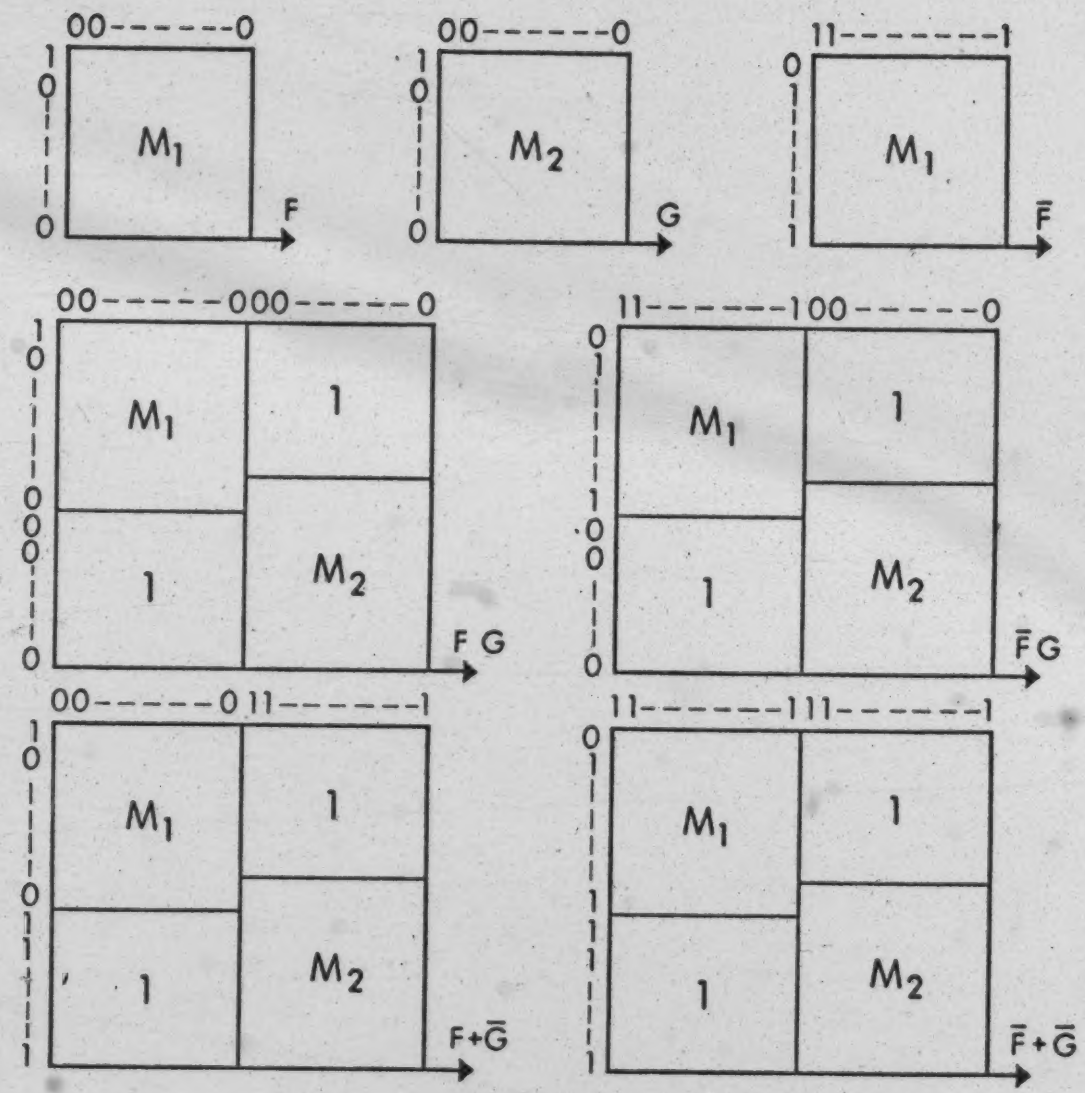
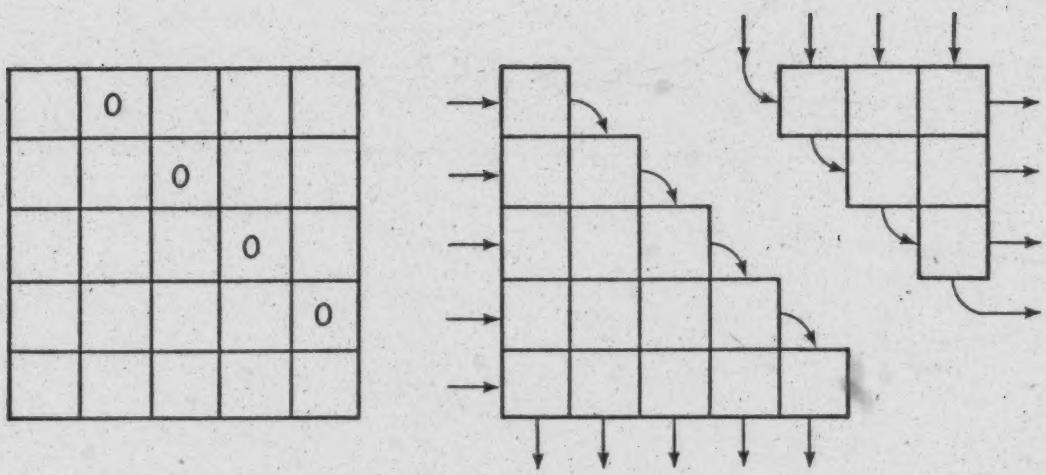
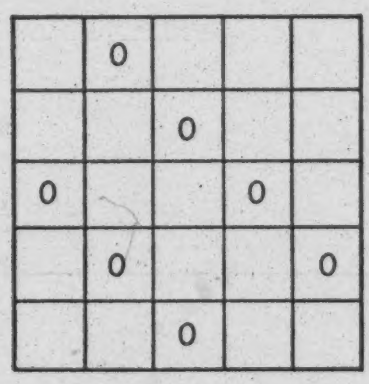


FIG. 13. INTERCONNECTING OF ARRAYS



(a) DIVIDED INTO TWO



(b) DIVIDED INTO THREE

FIG. 14. DIVIDING AN ARRAY

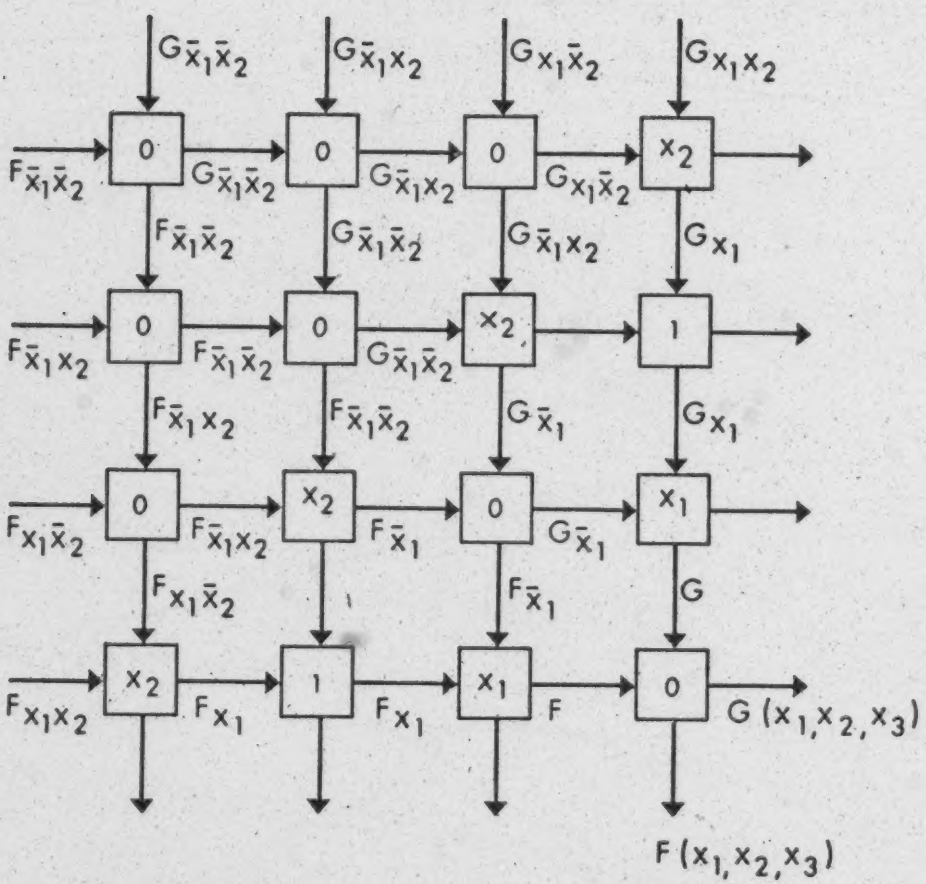


FIG. 15. REALIZATION OF TWO FUNCTIONS ON A CELLULAR PLANE

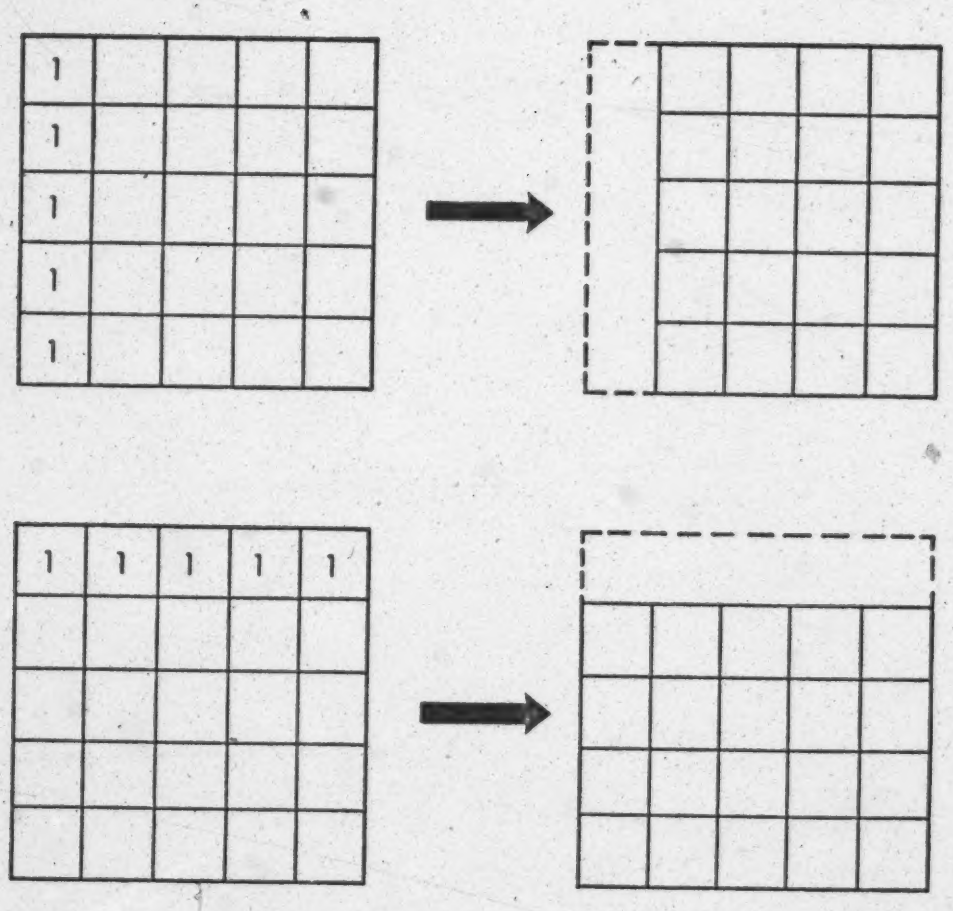


FIG. 16. STRIPPING AN ARRAY

E. Universal Logic Arrays

(1) Fixed Inputs

Any combinational switching function can be decomposed along its variables. For example, a function F of two variables C and D can be expressed as

$$F(C,D) = CF_C + \bar{C}F_{\bar{C}}$$

where

$$F_C = F[1,D]$$

$$F_{\bar{C}} = F[0,D]$$

F_C and $F_{\bar{C}}$ are the reduced functions of F with respect to C and $\bar{C}$ respectively. If $\alpha_1 = \bar{F}_C$ and $\alpha_2 = \bar{F}_{\bar{C}}$, then the array shown in fig. 17 is a universal array for two variables. Note that the inputs to the cellular plane are fixed and independent of the function being synthesized.

A universal array for three variables can be built by annexing two universal arrays of two variables. Let

$$F(B,C,D) = BF_B + \bar{B}F_{\bar{B}}$$

and

$$\alpha_1 = \bar{F}_{BC}$$

$$\alpha_2 = \bar{F}_{B\bar{C}}$$

$$\alpha_3 = \bar{F}_{\bar{B}\bar{C}}$$

$$\alpha_4 = \bar{F}_{\bar{B}C}$$

Note that $\alpha_I = 0, 1, \bar{D}$ or D for I = 1,2,3,4. Thus, any function F of three variables B,C and D can be realized by the cellular plane shown in fig. 18.

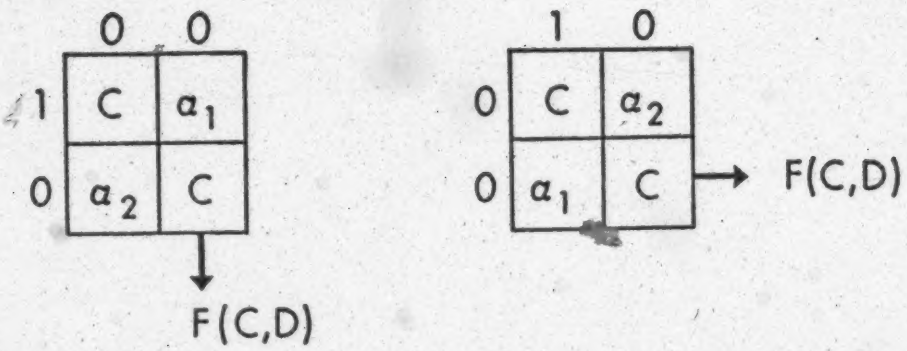


FIG. 17. UNIVERSAL ARRAYS FOR TWO VARIABLES C AND D

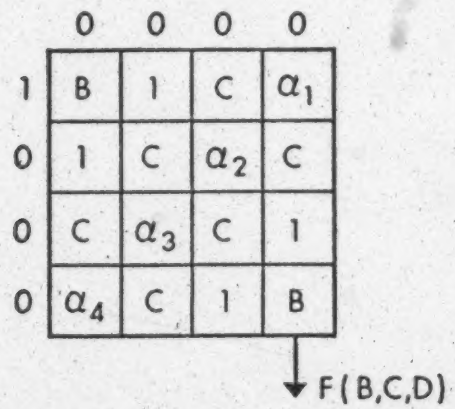


FIG. 18. UNIVERSAL ARRAY FOR 3 VARIABLES B, C and D

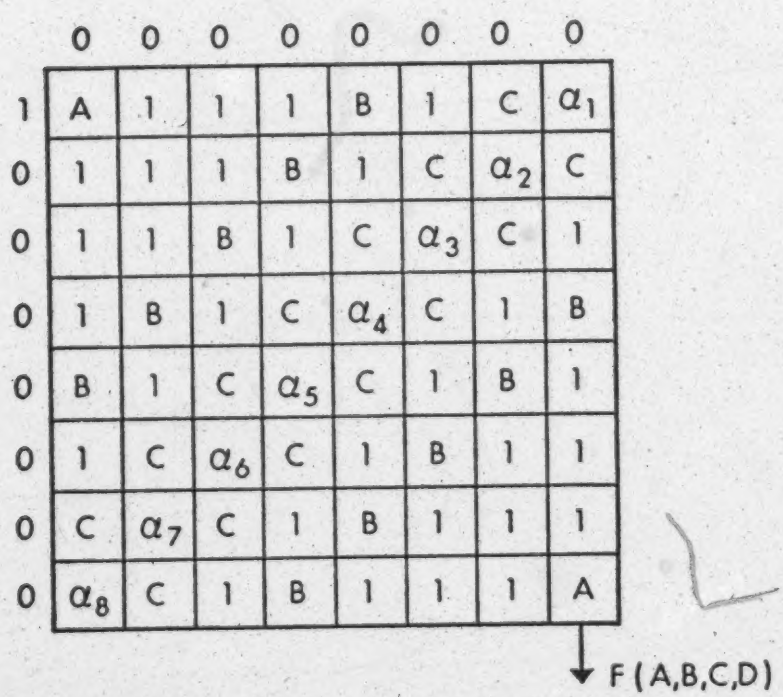

0 0 0 0 0 0 0 0
1 A 1 1 1 B 1 C α_1
0 1 1 1 B 1 C α_2 C
0 1 1 B 1 C α_3 C 1
0 1 B 1 C α_4 C 1 B
0 B 1 C α_5 C 1 B 1
0 1 C α_6 C 1 B 1 1
0 C α_7 C 1 B 1 1 1
0 α_8 C 1 B 1 1 1 A
F(A,B,C,D)

FIG. 19. UNIVERSAL ARRAY FOR 4 VARIABLES A, B, C, AND D

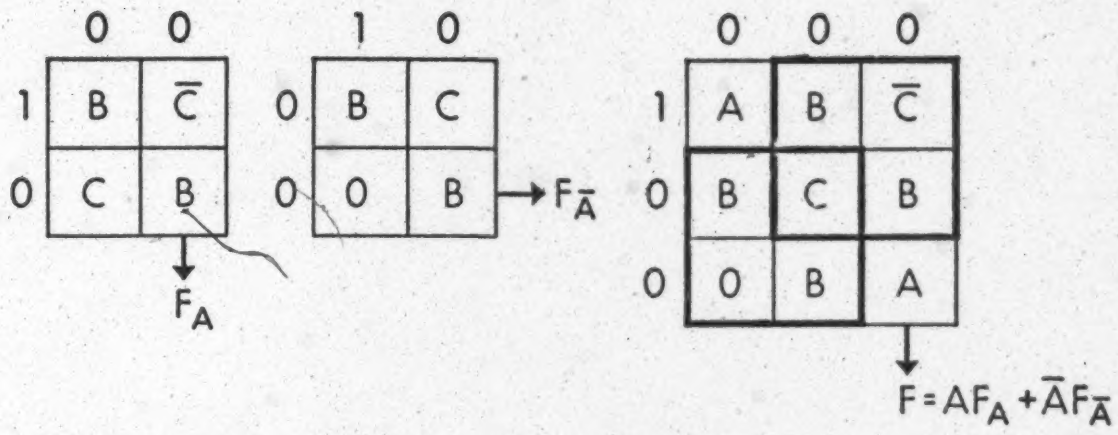
A similar argument leads to a universal logic array for four variables as shown in fig. 19. The array is composed of two universal logic arrays for three variables.

This method of constructing universal logic arrays can be extended recursively to functions of n variables. In general, any Boolean function of n variables can be realized by a cellular plane of dimensions $2^{n-1} \times 2^{n-1}$. Reduction of array size is possible if the upper right corner of the array realizing F_{x_I} and the lower left corner of the array realizing $F_{\bar{x}_I}$ can be overlapped, or the sizes of the arrays realizing F_{x_I} and/or $F_{\bar{x}_I}$ already have been reduced. Fig. 20 illustrates some possible cases.

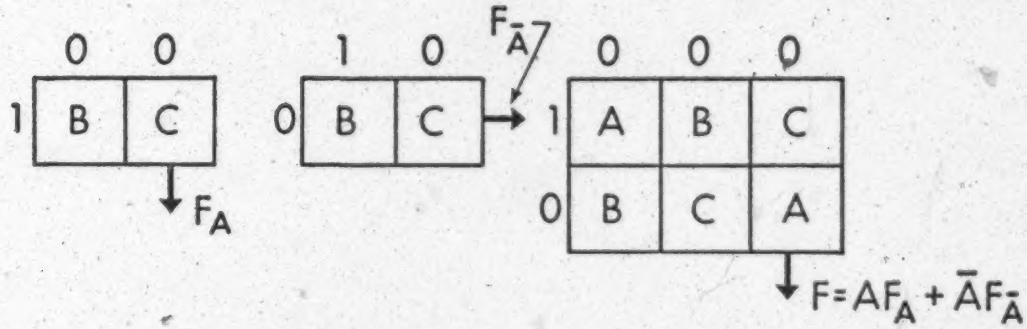
(II) Fixed Control Variables

So far in our universal logic arrays, we allow only the controls to the diagonal cells to change for realizing different functions. It is possible to keep the control variables fixed and vary the inputs to the edges of the array to realize different functions. Thus we obtain another class of useful array configurations. Fig. 21 shows the universal arrays for three and four variables by the fixed control variable method.

This method can be extended easily to n variables. The inputs to the cellular plane, which are the reduced functions of F with respect to n-1 variables, are ordered by appropriate choice of control variables. If M is a universal logic array for n-1 variables $x_1, x_2, \ldots, x_{n-1}$, then the universal logic array for n variables $x_1, x_2, \ldots, x_n$ is constructed in the manner shown in fig. 22. In general,



CASE 1: TWO SUB-ARRAYS OVERLAPPED



CASE 2: SUB-ARRAYS ALREADY REDUCED

FIG. 20. REDUCTION OF ARRAY SIZE

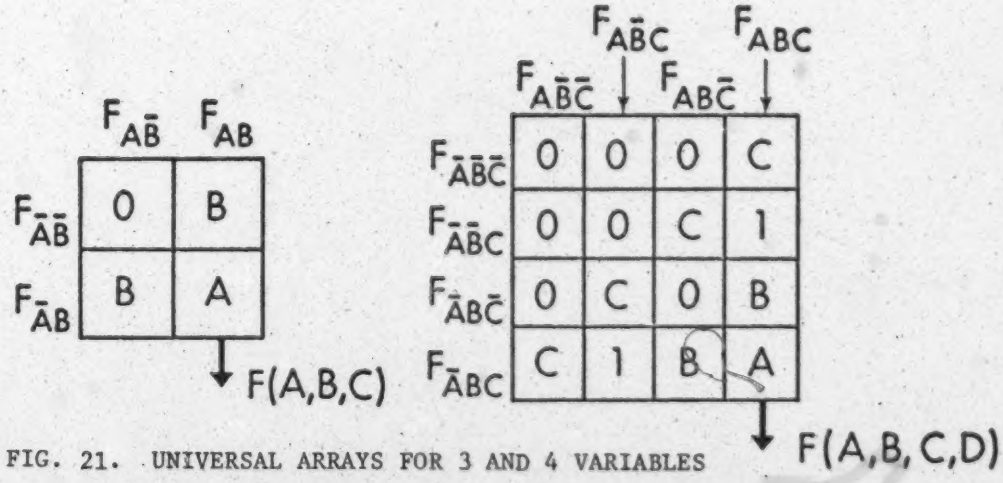


FIG. 21. UNIVERSAL ARRAYS FOR 3 AND 4 VARIABLES

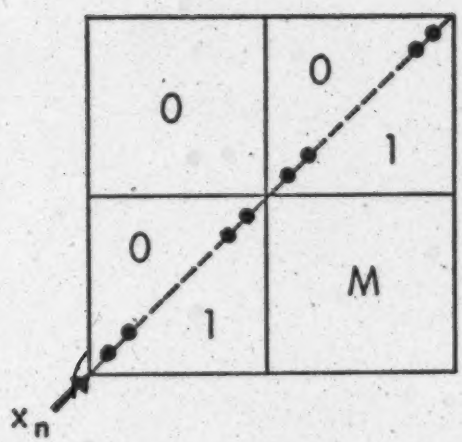


FIG. 22. n VARIABLES UNIVERSAL ARRAY BUILT FROM UNIVERSAL ARRAY FOR n-1 VARIABLES

an n-variable universal array will have dimensions $2^{n-2} \times 2^{n-2}$ and total size of 2^{2n-4} cells.

(III) Edge Fed Arrays

Some edge fed arrays [24] can be easily derived from the fixed control variable method. The cell circuit remains almost the same. The control variable is fed to the cell diagonally (figs. 23 and 24). We will consider two modified versions of the basic cell.

The first modified version of the basic cell is shown in fig. 23. The same figure also displays edge fed universal logic arrays for two, three and four variables. While the controls are fed along diagonal lines, the inputs to the cellular plane remain the reduced functions of F but appear in an unordered fashion. This technique of constructing universal logic arrays can be extended to any number of variables. In general, a universal logic array of n variables will require $2^{n-2} \times 2^{n-2}$ cells.

Another version of the basic cell which leads to useful edge fed array configuration is shown in fig. 24. Universal logic arrays for three and four variables are also illustrated in the same figure. This method of constructing universal logic arrays is not efficient for larger number of variables.

An interesting and efficient edge fed universal logic array for four variables is shown in fig. 25. Here F is given by

$$F = a_1[\bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD]$$
$$+ a_2[\bar{A}\bar{B}\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D}]$$
$$+ a_3[\bar{A}B\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D]$$

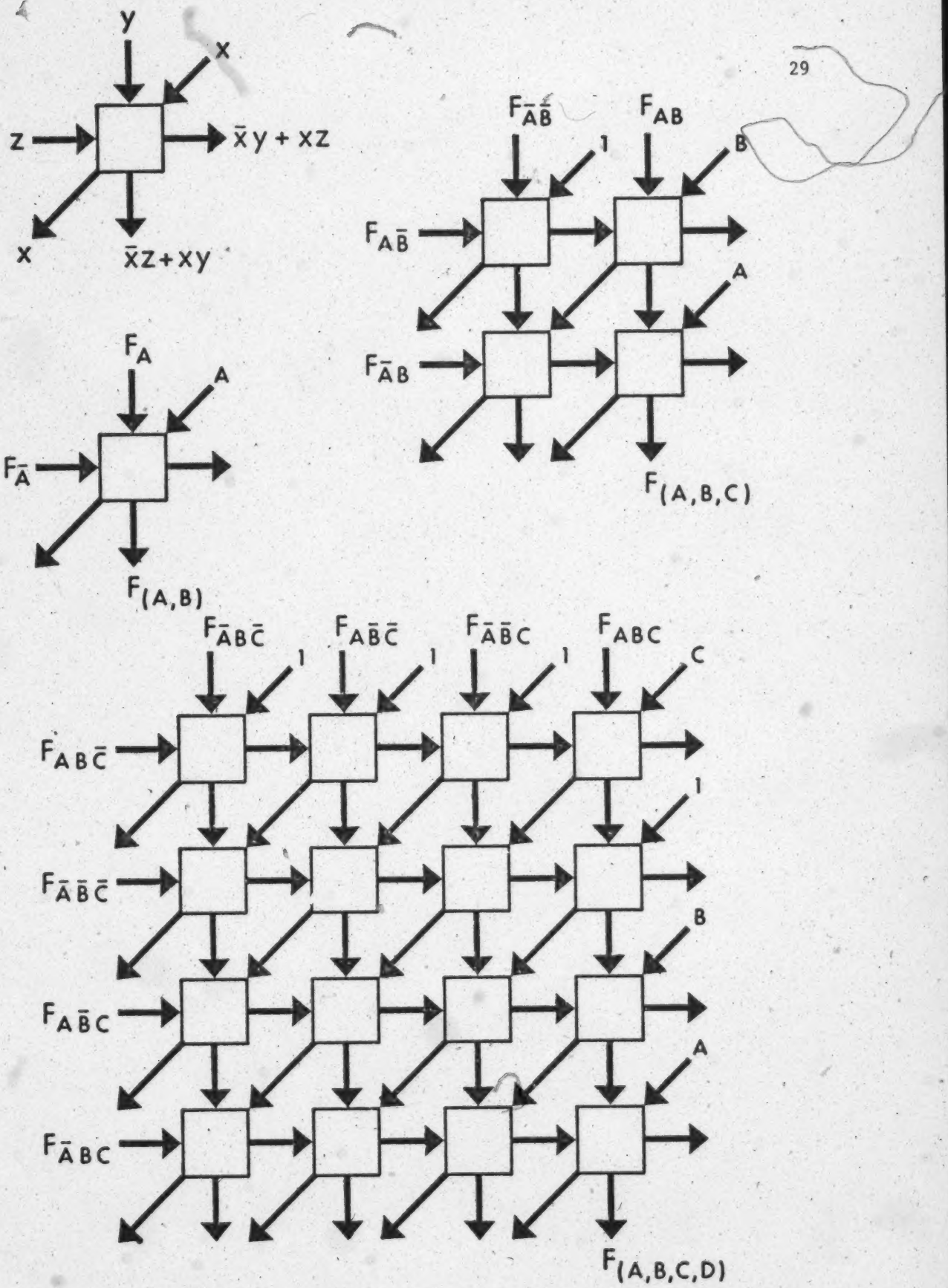


FIG. 23. A MODIFIED CELL MODEL, AND EDGE-FED UNIVERSAL ARRAYS FOR TWO, THREE, AND FOUR VARIABLES

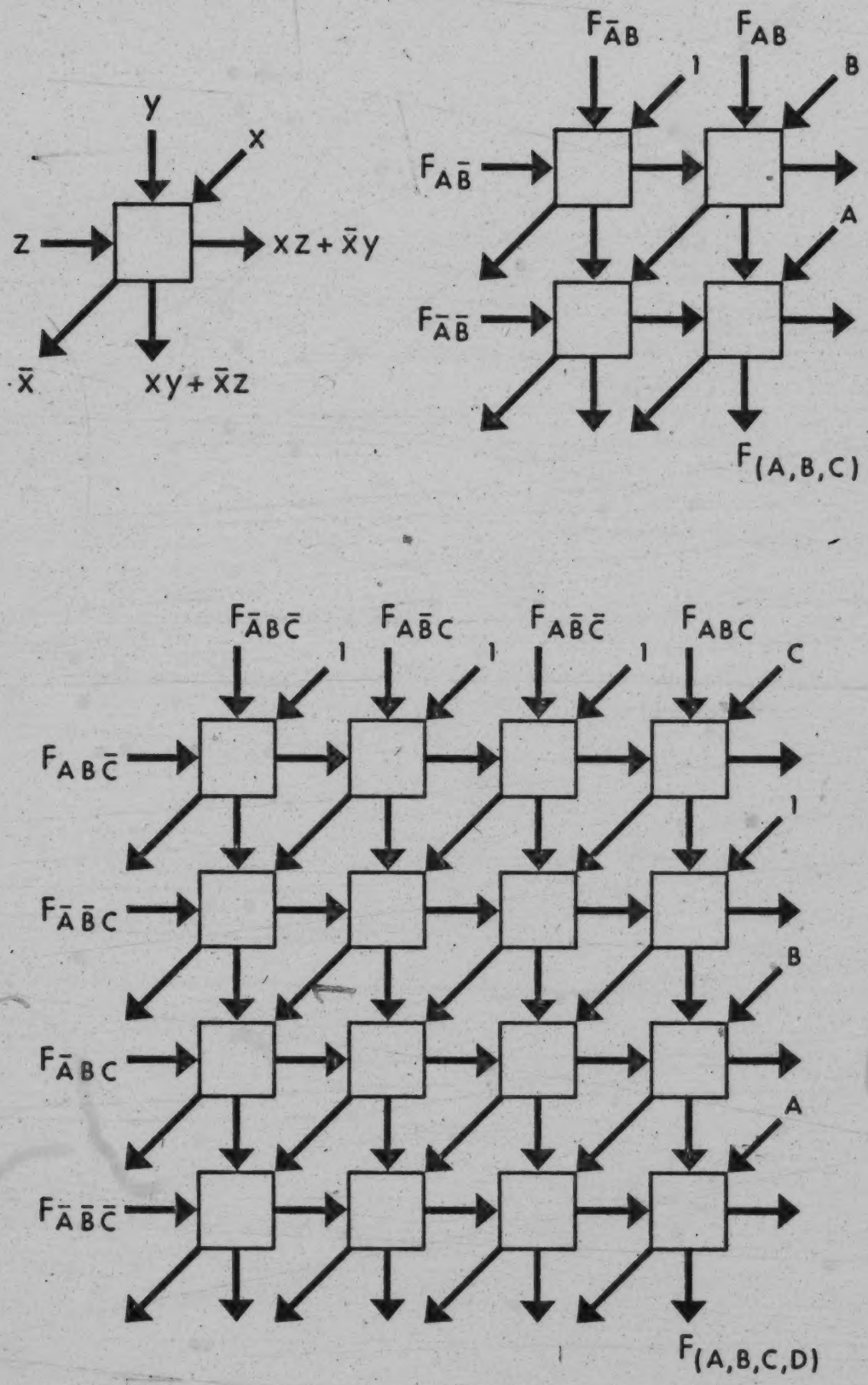


FIG. 24. OTHER VERSION OF CELL MODEL AND EDGE-FED UNIVERSAL ARRAYS

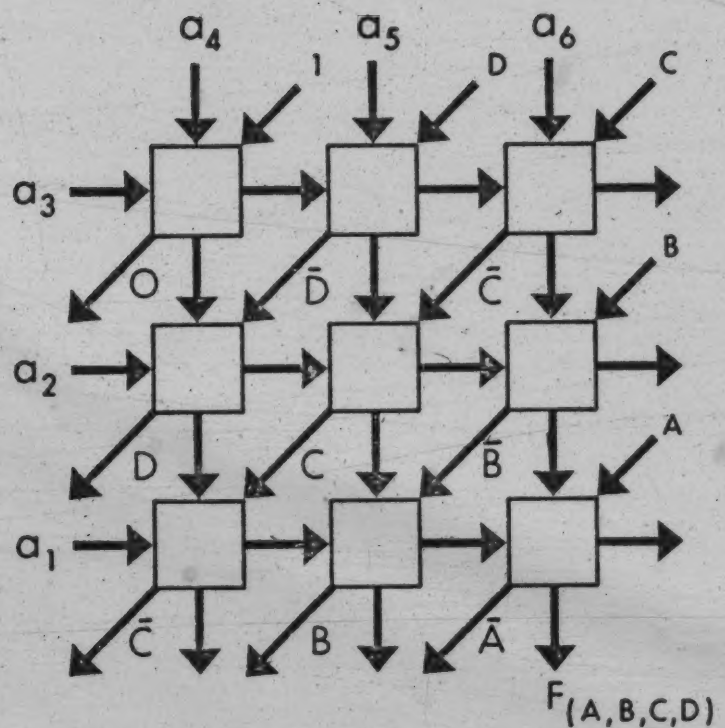


FIG. 25. A UNIVERSAL EDGE-FED ARRAY FOR FOUR VARIABLES

$$+ a_4[\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}D]$$
$$+ a_5[\bar{A}\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}\bar{D}]$$
$$+ a_6[ABC\bar{D} + ABCD]$$

Note that the expression for F contains all the vertices. Moreover, the vertices are divided into six disjoint sets V_1, V_2----,V_6. For input a_I (I = 1,2----,6) equal to 0,1 or some variable; none, all or a proper subset of the corresponding set V_I of the vertices can be selected. This technique of constructing efficient arrays fails for larger number of variables.

The edge fed arrays can also be implemented using the basic cell of section 2.1.

2.3 Cubic Cellular Arrays

A cubic cellular array is composed of identical cellular planes packed one upon the other such that all controls applied to the first plane will penetrate to all other planes without time delay. This feature enables us to realize a set of functions over the same arguments at the same time. This property will later be used in the realization scheme for an arbitrary synchronous sequential machine (Chapter III). Here we will describe some other interesting features of a cubic array when used as a combinational network.

A. Universal Cubic Array

The fixed control variable method described previously can be used to realize a set of combinational switching functions on a universal cubic array. Each cellular plane of the cubic array will

synthesize one function. Thus any K functions of the same n variables can be realized with a cubic array of dimensions K X 2^{n-2} X 2^{n-2}.

B. S-O-P and P-O-S Synthesis

Sometimes a logical designer is faced with a problem of realizing a function of (say) twenty variables. It may become tedious to find all the vertices in the function. Instead the function may be expressed in sum of products (S-O-P) or product of sums (P-O-S) form and then realized by an array. We will use the cubic array to realize a function expressed in either form. Each plane of the cube will synthesize a product or a sum term. Fig. 26 illustrates the realization of a function expressed in sum of products form. The function F synthesized is

$$F = ABCD + EFIJ + A\bar{B}GH$$

In fig. 26, the first plane realized $F_1 = \overline{ABCD}$ at its uppermost row. The outputs from the rest of the plane are neglected. The first row of the second plane is stripped (see 2.2D) by applying input 1 to the left edge of the left-most cell and to all the cells on the top edge. The term $F_2 = \overline{EFIJ}$ can then be synthesized from the second row of the plane. A similar argument holds for the third plane. The summation is performed over the fourth row of the fourth plane. Realization of a function $F = (\bar{A}+\bar{B}+\bar{C}+\bar{D})(\bar{E}+\bar{F}+\bar{I}+J)(\bar{A}+B+\bar{G}+\bar{H})$ expressed in product of sums form is shown in fig. 27.

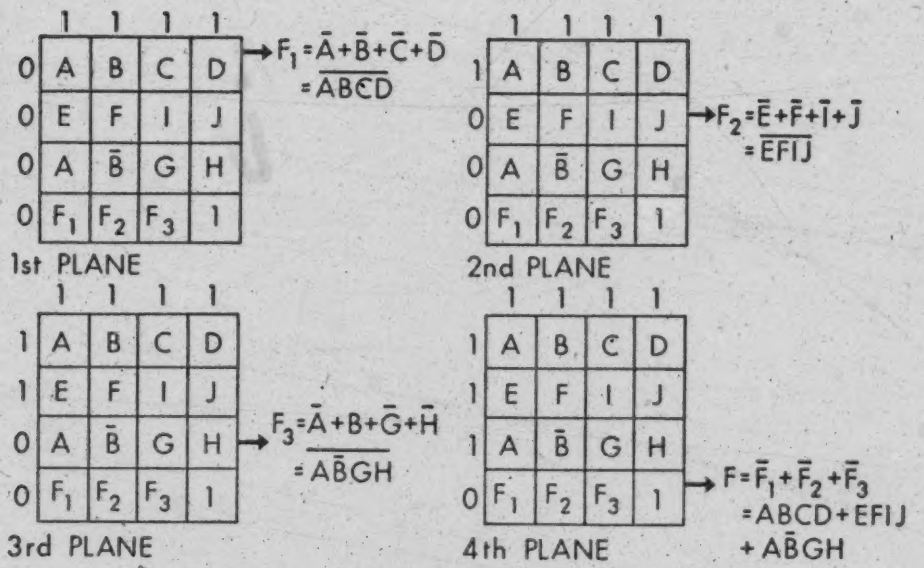


FIG. 26. CUBIC ARRAY REALIZING SUM-OF-PRODUCTS

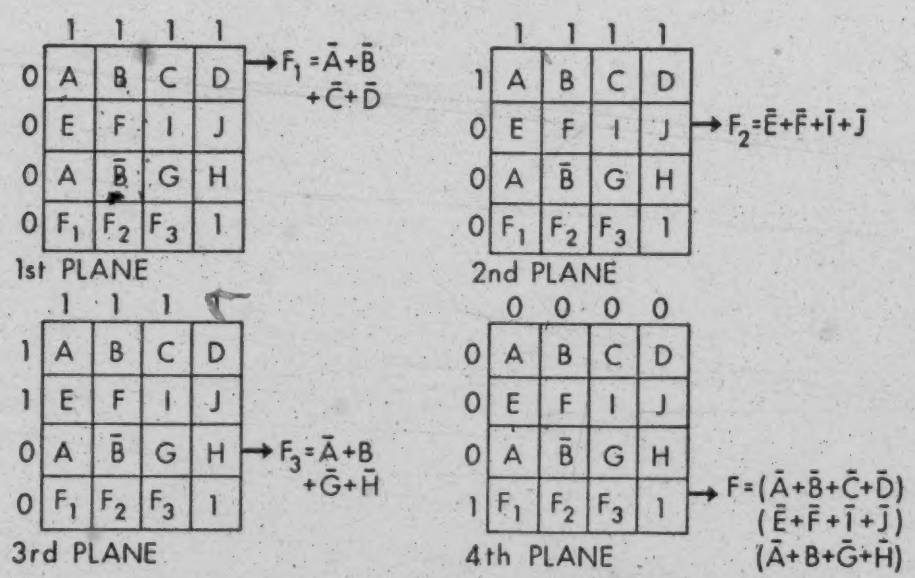


FIG. 27. CUBIC ARRAY REALIZING PRODUCT-OF-SUMS

2.4 Fault-Testing

A major problem faced by LSI circuit designers is that of fault testing. A number of authors have investigated this problem [2,25,26]. The most common method to detect and locate the fault is to generate sensitized paths and then route the signals from one edge through the path to the other edge of the array. Correct functioning of all the cells of the array allows proper routing of the signals. However, if a cell is faulty on the path, the input signals will not be transferred correctly to the output lead. The path is then shortened by one cell at a time until the faulty cell is located. In this section we follow a similar technique developed by K.J. Thurber [25]. First we describe a procedure to test the array and then the consequences of failure of any cell. Once the faulty cell is located, it can be isolated from the array by means of stripping process.

To test the array, it is sufficient to ensure proper operation of the two modes of all the cells. The paths through the top right corner cell are of minimum length. Thus it will be convenient to test the top right corner cell and then move from cells on one diagonal to the next diagonal. We will assume the array is of dimension (ℓ,m). The test procedure in the absence of faults is as follows:

Step 1: Set all controls of the cellular plane to 0. All the cells will be in bending mode.

Step 2: Vary the inputs to the cells (1,m) and (1,m-1), and observe the outputs at the right edges of the cells (1,m) and (2,m) respectively. Correct outputs indicate that the top right

corner cell is functioning properly in the bending mode. Change the control of the top right corner cell to 1 and repeat the test. This time, the output leads get interchanged, i.e., the output leads of the cells (1,m) and (2,m) become transparent to signals placed on input leads of the cells (1,m-1) and (1,m) respectively. This tests the crossing mode of the top right corner cell. Switch back the control of the top right corner cell to 0.

Step 3: To test the next diagonal cells (1,m-1) and (2,m), vary the inputs to the cells (1,m-2) and (1,m-1), and observe the respective outputs at the right edges of the cells (3,m) and (2,m). Repeat the test, changing the control of the diagonal cell to 1, one at a time. Every time the control of a diagonal cell is changed, the output leads get interchanged if the cells are functioning properly. Control of the diagonal cells can now be switched back to 0.

Step 4: This procedure continues for subsequent cells until the complete array is tested.

In the presence of failures, a particular path will not transfer correctly the input sequence to the output lead. We must conclude that at least one cell on the path is faulty. To locate the faulty cell the path will be shortened by one cell at a time. We will first make the following assumptions to simplify the procedure.

1. Any failures which occur are constant with time. The problem of intermittent failures is neglected.
2. Only one cell on the path is faulty and, moreover, the output of only one gate is stuck to 0 or 1.
3. No buses or interconnections have failed.

In Fig. 28, assume that a cell on the path PQRST is faulty. To locate that cell, first change the control of cell F to 1. Vary the input at P and observe the output sequence at the bottom edge of cell F. If the output sequence corresponds correctly to the input sequence, we conclude that the switch S_4 in cell F is faulty. An incorrect input/output correspondence indicates that either the switch S_1 in cell F is faulty or the input sequence to the cell F is itself incorrect. Now change the control of cell D to 0 and repeat the test, this time the output lead is the right edge of cell D. If input/output correspondence is still incorrect, cell F is perfect and we should proceed to test the path PQRS. If we observe a correct correspondence, we conclude either the switch S_1 in cell F or the switch S_3 in cell D is faulty. In this case, either of the two cells can be isolated from the array by the stripping process.

If the test reveals that a cell C_{IJ} should be isolated from the array, a fixed logical 1 control is applied to all cells on the I-th row and J-th column. This strips off the I-th row and the J-th column and results in a smaller array (fig. 29).

The above procedure can also be used to locate multiple faults on a path. We will assume that the output of only one gate is

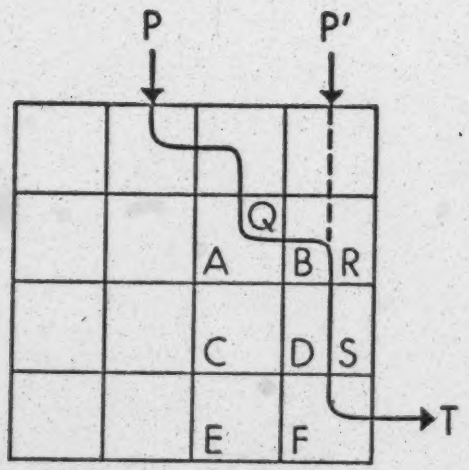


FIG. 28. DETECTING A FAULTY CELL ON A PATH

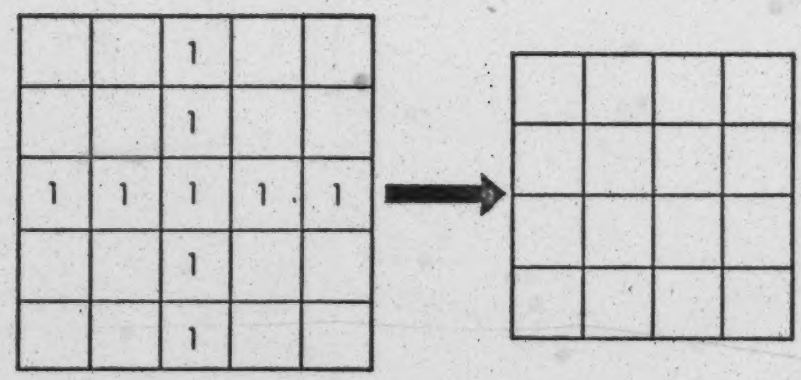


FIG. 29. ISOLATING A CELL

stuck to 0 or 1 in any of the faulty cell on the path. Since the path is directed, the test procedure will locate the fault nearest to the origin of the path. For example, in fig. 28, assume that cells A and F are faulty on the path PQRST. The test procedure will indicate that cell A is not functioning properly in its bending mode. All the cells on the row and column through this cell are set to control 1. This isolates the defective cell from the array. The path $\acute{P}RT$ is now constructed by applying proper controls. The test procedure now applied to the path $\acute{P}RT$ will detect the fault in cell F. The cell can then be isolated by the stripping process.

2.5 Comparison

In this section we will compare our array with different cellular structures proposed by other authors. The comparison could be based on various design parameters such as growth rate, the number of cutpoint inputs, the number of edge inputs to the array, complexity of the cell and interconnection structure and many other electrical and geometrical parameters compatible with a given technology [3]. Some of these parameters are listed below.

(I) Growth Rate

A typical expression for d(n) is

$$d(n) = n[C_o 2^n + C_1 2^{n-1} \text{----} + C_n]$$

where d(n) is the number of cells in a universal logic array of n variables, C_o, C_1---,C_n are all constants. As n increases, the number of cells becomes approximately $C_o n 2^n$. This latter quantity is

termed the growth rate. A typical growth rate for rectangular cellular arrays found by Minnick [1] is C n 2^n.

(II) Input Connections

Since the number of input terminals on an array strongly affects the cost of that array, it forms an important parameter in the comparison. Note that there could be fixed logical constant inputs to the array which are independent of the function being realized. These inputs can be provided at the time of fabrication of the array.

(III) Cutpoint Inputs

In all these arrays, we need external inputs to each cell to perform cutpoint operation. Since the cost of implementing such an operation depends upon the number of cutpoint inputs, it must also be considered in the comparison.

(IV) Inter-Cell Connections

Another factor which affects the cost of implementing these arrays on LSI is the number of inter-cell connections. A realistic measure for this is the number of paths into or out of a cell.

With these parameters under consideration, the combinational cellular arrays of this chapter are compared with those proposed by other authors [Table I].

TABLE I

Comparison of Cellular Arrays

	Minnick's cut-point Array [1]	Kautz's q-Function Array [3]	Short's Majority Gate Arrays [1]		Aker's Array [27]	Our Array		
			First Version	Second Version		Fixed Input	Fixed Control Variable	Edge Fed
Growth Rate	$n\ 2^{n-1}$	$n\ 2^{n-1}$	2^{2n-3}	$3n\ 2^{n}$	2^{2n-4}	2^{2n-2}	2^{2n-4}	2^{2n-4}
Input Connections	$3(n+1)\ 2^{n-1} + 2^{n-1} + n+1$	$\frac{1}{3}(n+1)(2^{n+1}+1)+ \frac{1}{3}(2^{n+1}+1) + n+1$	$2^{2n-3} + 2^{n-1}$	---	$2^{2n-4} + 2^{n}+3$	$2^{2n-2} + 2^{n}$	$2^{2n-4} + 2^{n-1}$	2^{n}
Inter-Cell Connections	2	2	2	2	2	2	2	3
Cutpoint Inputs/Cell	3	2	2	2	2	2	2	0

CHAPTER 3

SEQUENTIAL CELLULAR STRUCTURES

This chapter deals with sequential cellular structures using the same basic cell [30]. The synthesis procedure is analogous to the one developed by Arnold et al. [19-21]. Here we will not discuss their realizations but point out the basic difference. In their scheme, all the modules are delayed logic modules and the resulting realization is in the form of a tree pattern. To realize sequential machines with multiple inputs and/or feedback functions, the modules are modified by addition of more inputs and more combinational circuitry. However, the modules still remain delayed logic modules. In our module, the delay can be introduced or cut off by setting the value of control variable T. It will be interesting to see how this simple cell structure can be used to realize an arbitrary synchronous sequential machine. For simplicity, we will start with binary input - binary output machines and then extend the technique to machines with multiple inputs and/or feedback functions.

3.1 Definite Machines

Given a definite machine M_1 with state table shown below, we note that the output is the set of states S_4, S_6 and S_7. Let S_{ijk} denote the set (S_i, S_j, S_k) and $S_{ijk}(t)$ denote that the machine is in state S_i, S_j or S_k at time t.

From the state table of M_1, we find that the machine enters the state set S_{467} if it was previously in S_{36} with present input 0 or in S_{3456} with present input 1, i.e.

$$Z(t) = x(t)S_{3456}(t-1) + \bar{x}(t)S_{36}(t-1)$$

Table 2

State Table of M_1

S \ x	0	1	z
1	1	3	0
2	5	2	0
3	4	7	0
4	1	6	1
5	1	6	0
6	4	7	1
7	5	2	1

Expanding the sets S_{3456} and S_{36} in terms of the previous sets, we have

$$S_{36}(t) = \bar{x}(t)S_{\phi}(t-1) + x(t)S_{145}(t-1)$$

$$S_{3456}(t) = \bar{x}(t)S_{2367}(t-1) + x(t)S_{145}(t-1)$$

and

$$S_{145}(t) = \bar{x}(t)S_{I}(t-1) + x(t)S_{\phi}(t-1)$$

$$S_{2367}(t) = \bar{x}(t)S_{\phi}(t-1) + x(t)S_{I}(t-1)$$

where S_{ϕ} is the null set and S_{I} is the set of all states of the machine. A realization of M_1 is given in Fig. 30, where S_{ϕ} is represented by a logical 0 and S_{I} by a logical 1.

The synthesis procedure described above can be applied to any definite machine. In general, an arbitrary binary input - binary output definite machine can be realized using one cellular plane of the cube. Moreover, for an n-state machine, since the memory is at most n-1, an upper bound to the number of cells required is $2^{n-1} \times 2^{n-1}$.

3.2 Regular Machines

An arbitrary binary input - binary output synchronous sequential machine will in general require some sort of feedback. The realization scheme may employ either a single feedback function with finite memory or multiple feedback functions with unit memory [23]. These realization schemes will be illustrated with the help of an example.

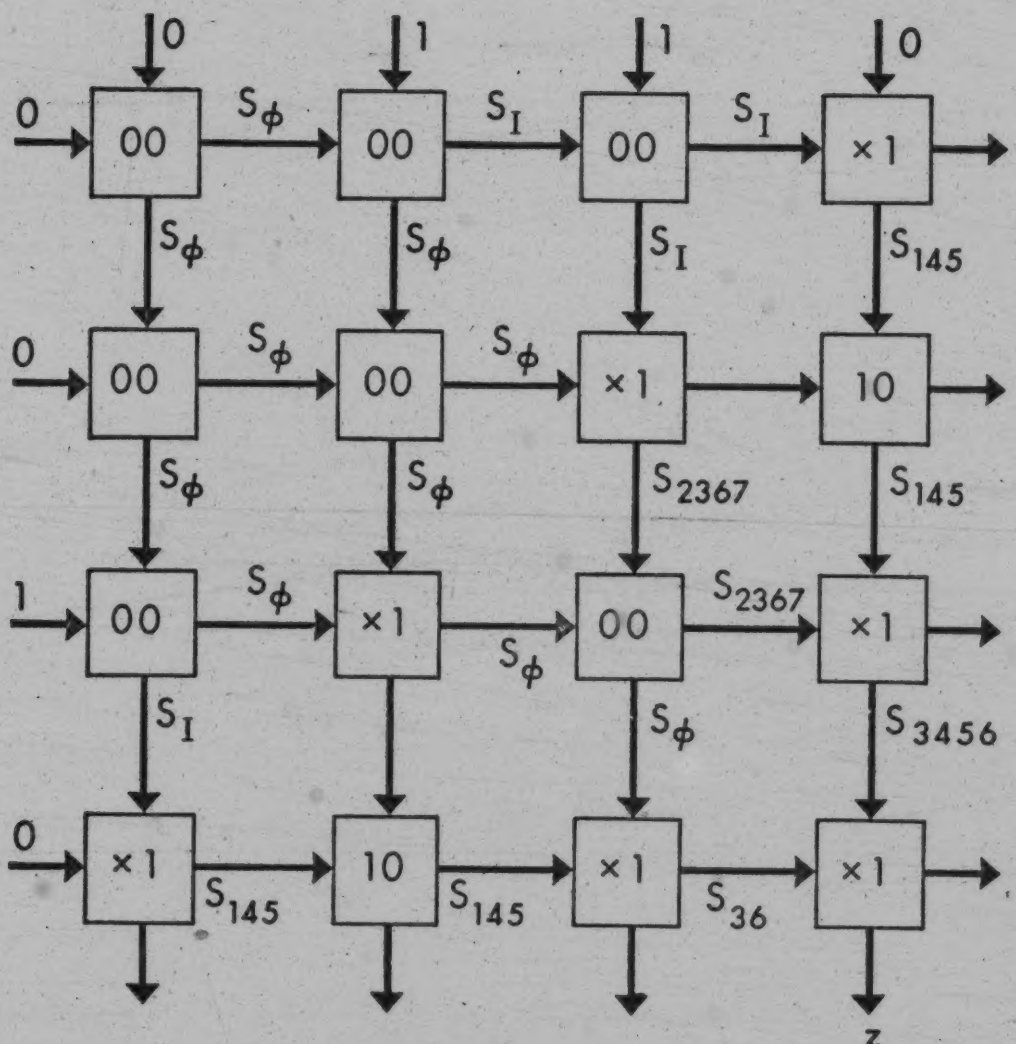


FIG. 30 REALIZATION OF M_1

(I) Single Feedback Realization

The machine M_2 is here used as an illustrative example. The machine happens to have finite memory with respect to input and output. Thus the output function z can be used as a feedback function. The state table and expanded flow table [23] for M_2 are shown below.

Table 3

State Table and Expanded Flow Table of M_2 Using Single Feedback Realization Scheme

State Table of M_2

S \ x	0	1	z
1	2	1	1
2	2	4	1
3	4	3	0
4	1	3	0

Expanded Flow Table of Machine M_2

S \ xz	00	01	11	10
1	-	2	1	-
2	-	2	-	4
3	4	-	-	3
4	-	1	-	3

Using the technique of Arnold et al [20], we have

$$z(t) = S_{12}(t) = \bar{x}(t)S_{124}(t-1) + x(t)S_1(t-1)$$

Let S^*_{ijk} denote that S_i, S_j, and S_k are DON'T CARE entries. Representing the sets S_1, S_{124} in terms of the previous sets, we have

$$S_1(t) = \bar{x}(t)\bar{z}(t)[S_\phi(t-1) + S^*_{124}(t-1)] +$$
$$\bar{x}(t)z(t)[S_4(t-1) + S^*_3(t-1)] +$$
$$x(t)\bar{z}(t)[S_\phi(t-1) + S^*_1(t-1)] +$$
$$x(t)z(t)[S_1(t-1) + S^*_{234}(t-1)]$$

$$S_{124}(t) = \bar{x}(t)\bar{z}(t)[S_3(t-1) + S^*_{124}(t-1)] +$$
$$\bar{x}(t)z(t)[S_{124}(t-1) + S^*_3(t-1)] +$$
$$x(t)\bar{z}(t)[S_2(t-1) + S^*_1(t-1)] +$$
$$x(t)z(t)[S_1(t-1) + S^*_{234}(t-1)]$$

Choosing the DON'T CARE entries so that it will make the previous sets either equal to $S_I(t-1)$ or $S_\phi(t-1)$, we then have

$$S_1(t) = \bar{x}(t)\bar{z}(t)S_\phi(t-1) + \bar{x}(t)z(t)[s_4(t-1) + S^*_3(t-1)] +$$
$$x(t)\bar{z}(t)S_\phi(t-1) + x(t)z(t)S_I(t-1)$$

$$S_{124}(t) = \bar{x}(t)\bar{z}(t)S_I(t-1) + \bar{x}(t)z(t)S_I(t-1) +$$
$$x(t)\bar{z}(t)[S_2(t-1) + S^*_1(t-1)] + x(t)z(t)S_I(t-1)$$

Successively, we have

$$S_4(t) + S^*_3(t) = \bar{x}(t)\bar{z}(t)[S_3(t-1) + S^*_{124}(t-1)] +$$
$$\bar{x}(t)z(t)[S_\phi(t-1) + S^*_3(t-1)] +$$
$$x(t)\bar{z}(t)[S_2(t-1) + S^*_{134}(t-1)] +$$
$$x(t)z(t)[S_\phi(t-1) + S^*_{234}(t-1)]$$
$$= \bar{x}(t)\bar{z}(t)S_I(t-1) + \bar{x}(t)z(t)S_\phi(t-1) +$$
$$x(t)\bar{z}(t)S_I(t-1) + x(t)z(t)S_\phi(t-1)$$

$$S_2(t) + S^*_1(t) = \bar{x}(t)\bar{z}(t)[S_\phi(t-1) + S^*_{124}(t-1)] +$$
$$\bar{x}(t)z(t)[S_{12}(t-1) + S^*_{34}(t-1)] +$$
$$x(t)\bar{z}(t)[S_\phi(t-1) + S^*_1(t-1)] +$$

	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	x1
1	00	••	••	••	••	••	••	••	••	••	••	••	••	00	x1	10
1	00	••	••									••	00	x1	10	10
1	00	••	••								••	00	x1	10	••	10
0	00	••	••			00				••	00	x1	10	••	••	10
0	00	••	••						••	00	x1	10	••	10	••	10
1	00	••	••					••	00	x1	10	••	••	••	••	10
0	00	••	••				••	00	x1	10	10	10	10	10	10	10
1	00	••	••			••	00	x1	00	00	00	00	00	00	00	z0
1	00	••	••		••	00	x1	10	00	••	••	••	••	00	z0	10
1	00	••	••	••	00	x1	10	10	00	••	00	••	00	z0	10	10
1	00	••	••	00	x1	10	••	10	00	••	••	00	z0	10	10	10
1	00	••	00	x1	10	••	••	10	00	••	00	z0	00	00	00	x1
1	00	00	x1	10	••	10	•••	10	00	00	z0	10	00	00	x1	10
1	00	x1	10	••	••	••	••	10	00	z0	10	10	00	x1	00	z0
1	x1	10	10	10	10	10	10	10	z0	10	10	10	x1	10	z0	x1

z

FIG. 31. SINGLE FEEDBACK REALIZATION OF M_2

$$x(t)z(t)[S_\phi(t-1) + S_I^*(t-1)]$$

$$= \bar{x}(t)\bar{z}(t)S_\phi(t-1) + \bar{x}(t)z(t)S_I(t-1) +$$

$$x(t)\bar{z}(t)S_\phi(t-1) + x(t)z(t)S_\phi(t-1).$$

A realization based on the above equations is shown in Fig. 31, where only one cellular plane is required. This is true for all finite memory machines. For machines which do not have finite memory, the feedback function must be properly chosen. In some cases, splitting up of states will be required, as discussed by Friedman [23]. The realization will then require two cellular planes, one for the output and the other for the feedback function.

For single feedback realization, the memory μ has been found [23] to be

$$2\log_2 n + [\log_2(\log_2 n+1)] + 1$$

This puts an upper limit to the number of cells required of

$$2 \times 2^{2\mu-1} \times 2^{2\mu-1} = 8n^8 (\log_2 2n)^4.$$

where n is the number of states of the machine.

(II) Single Stage Realization

Another way of realizing a machine is to use more feedback functions such that the machine will have a memory of 1. We choose a set of feedback functions in such a way that the input together with feedback functions can identify the states of the machine in one stage. In this case, the output can be expressed as a combinational function of the present input and the previous values of the input and feedback

functions. Machine M_2 will again be used for illustration. The feedback function f together with the output function z can identify each state of the machine. The state table and expanded flow table are shown below.

Table 4

State Table and Expanded Flow Table of M_2 Using Single Stage Realization Scheme

State Table of M_2

S \ x	0	1	- zf
1	2	1	11
2	2	4	10
3	4	3	00
4	1	3	01

Expanded Flow Table of M_2

S \ xzf	000	001	010	011	100	101	110	111
1	-	-	2	-	-	-	-	1
2	-	-	2	-	-	4	-	-
3	-	4	-	-	3	-	-	-
4	-	-	-	1	3	-	-	-

Using the same technique as in single feedback realization, we have

$$Z(t) = S_{12}(t) = \bar{x}(t)S_{124}(t-1) + x(t)S_1(t-1)$$

$$f(t) = S_{14}(t) = \bar{x}(t)S_{34}(t-1) + x(t)S_{12}(t-1)$$

$$\begin{aligned} S_{124}(t) = {} & \bar{x}(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) + \\ & \bar{x}(t)\bar{z}(t)f(t)[S_3(t-1) + S^*_{124}(t-1)] + \\ & \bar{x}(t)z(t)\bar{f}(t)[S_{12}(t-1) + S^*_{34}(t-1)] + \\ & \bar{x}(t)z(t)f(t)[S_4(t-1) + S^*_{124}(t-1)] + \\ & x(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) + \\ & x(t)\bar{z}(t)f(t)[S_2(t-1) + S^*_{134}(t-1)] + \\ & x(t)z(t)\bar{f}(t)S_{\phi}(t-1) + \\ & x(t)z(t)f(t)[S_1(t-1) + S^*_{234}(t-1)] \end{aligned}$$

$$\begin{aligned} S_1(t) = {} & \bar{x}(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) + \\ & \bar{x}(t)\bar{z}(t)f(t)S_{\phi}(t-1) + \\ & \bar{x}(t)z(t)\bar{f}(t)S_{\phi}(t-1) + \\ & \bar{x}(t)z(t)f(t)[S_4(t-1) + S^*_{123}(t-1)] + \\ & x(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) + \\ & x(t)\bar{z}(t)f(t)S_{\phi}(t-1) + \\ & x(t)z(t)\bar{f}(t)S_{\phi}(t-1) + \\ & x(t)z(t)f(t)[S_1(t-1) + S^*_{234}(t-1)] \end{aligned}$$

$$S_{34}(t) = \bar{x}(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$\bar{x}(t)\bar{z}(t)f(t)[S_3(t-1) + S^*_{124}(t-1)] +$$

$$\bar{x}(t)z(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$\bar{x}(t)z(t)f(t)S_{\phi}(t-1) +$$

$$x(t)\bar{z}(t)\bar{f}(t)[S_{34}(t-1) + S^*_{12}(t-1)] +$$

$$x(t)\bar{z}(t)f(t)[S_2(t-1) + S^*_{134}(t-1)] +$$

$$x(t)z(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$x(t)z(t)f(t)S_{\phi}(t-1)$$

$$S_{12}(t) = \bar{x}(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$\bar{x}(t)\bar{z}(t)f(t)S_{\phi}(t-1) +$$

$$\bar{x}(t)z(t)\bar{f}(t)[S_{12}(t-1) + S^*_{34}(t-1)] +$$

$$\bar{x}(t)z(t)f(t)[S_4(t-1) + S^*_{123}(t-1)] +$$

$$x(t)\bar{z}(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$x(t)\bar{z}(t)f(t)S_{\phi}(t-1) +$$

$$x(t)z(t)\bar{f}(t)S_{\phi}(t-1) +$$

$$x(t)z(t)f(t)[S_1(t-1) + S^*_{234}(t-1)]$$

Wherever possible, the DON'T CARE entries are used or neglected to make the previous set S_I or S_{ϕ} respectively. The realization of M_2 based on these equations is shown in Fig. 32. Note that we require two cellular planes of the cube to realize z and f.

	0	0	0	0	0	1	0	1
0	00	00	00	00	00	00	00	x1
0	00	00	00	00	00	00	x1	10
1	00	00	00	00	00	x1	10	10
1	00	00	00	00	x1	10	10	10
1	00	00	00	x1	00	00	00	f0
0	00	00	x1	10	00	00	f0	10
1	00	x1	10	10	00	f0	00	z0
1	x1	10	10	10	f0	10	z0	x1

↓ z

	0	1	0	0	0	1	0	1
0	00	00	00	00	00	00	00	x1
0	00	00	00	00	00	00	x1	10
1	00	00	00	00	00	x1	10	10
0	00	00	00	00	x1	10	10	10
1	00	00	00	x1	00	00	00	f0
0	00	00	x1	10	00	00	f0	10
1	00	x1	10	10	00	f0	00	z0
0	x1	10	10	10	f0	10	z0	x1

↓ f

FIG. 32. SINGLE STAGE REALIZATION OF M_2

In general, for a machine of n states, the number of feedback functions is at most m, where $2^{m-1} < n \leq 2^m$. Thus, any binary input - binary output synchronous sequential machine can be realized with a cubic array of (m+1) planes. The number of cells in each plane will be

$$2^{m+1} \times 2^{m+1} = 2^{2m+2} \leq 16n^2$$

An upper bound to the number of cells required will be

$$(m+1) \times 16n^2 = [\log_2 2n] \times 16n^2$$

where [x] means the smallest integer larger than or equal to x.

3.3 Multiple Inputs, Outputs, and Feedback Variables

The technique used above can be extended to machines with multiple inputs, outputs, and feedback variables. Let a machine have inputs $x_1, x_2, \ldots, x_n$ and outputs $z_1, z_2 \ldots, z_m$. Let $f_1, f_2 \ldots, f_k$ be the feedback functions such that the machine has finite memory with respect to inputs and feedback functions. The synthesis procedure will be as follows:

(1) Express every output and feedback variable in terms of all possible combinations of inputs and previous state sets.

$$z_i(t) = \bar{x}_1(t)\text{----}\bar{x}_n(t)S_{i_1}(t-1) +$$
$$\bar{x}_1(t)\text{----}x_n(t)S_{i_2}(t-1) +$$
$$\vdots$$
$$x_1(t)\text{----}x_n(t)S_{i_{2^n}}(t-1)$$

(2) Express every S_{i_j} obtained in the previous step in terms of all possible combinations of inputs, feedback variables, and previous state sets (except those which are S_ϕ or S_I already).

$$S_{i_j} = \bar{x}_1(t)\text{----}\bar{x}_n(t)\bar{f}_1(t)\text{----}\bar{f}_k(t)S_{i_{j_1}}(t-1) +$$
$$\bar{x}_1(t)\text{----}\bar{x}_n(t)\bar{f}_1(t)\text{----}f_k(t)S_{i_{j_2}}(t-1) +$$
$$\vdots$$
$$x_1(t)\text{----}x_n(t)f_1(t)\text{----}f_k(t)S_{i_{j_r}}(t-1)$$

where $r = 2^{n+k}$.

For each $S_{i_{j_k}}$ in the above equation, if it is not S_ϕ or S_I, then it will contain the DON'T CARE entries.

(3) Repeat step (2) until all $S_{i_{j_k}}$ are either S_ϕ or S_I, for all i,j and k.

The above procedure will always terminate because the machine memory is finite. Fig. 33 shows the schematic realization of a sequential machine with n inputs, m outputs, and k feedback functions. Every output or feedback function is realized by one plane of the cube. The feedback functions are then used as part of the control variables to the array. In Fig. 33, inputs to the array are not shown because they vary from machine to machine. Constant control variables as well as 0 values of T are also omitted from the figure.

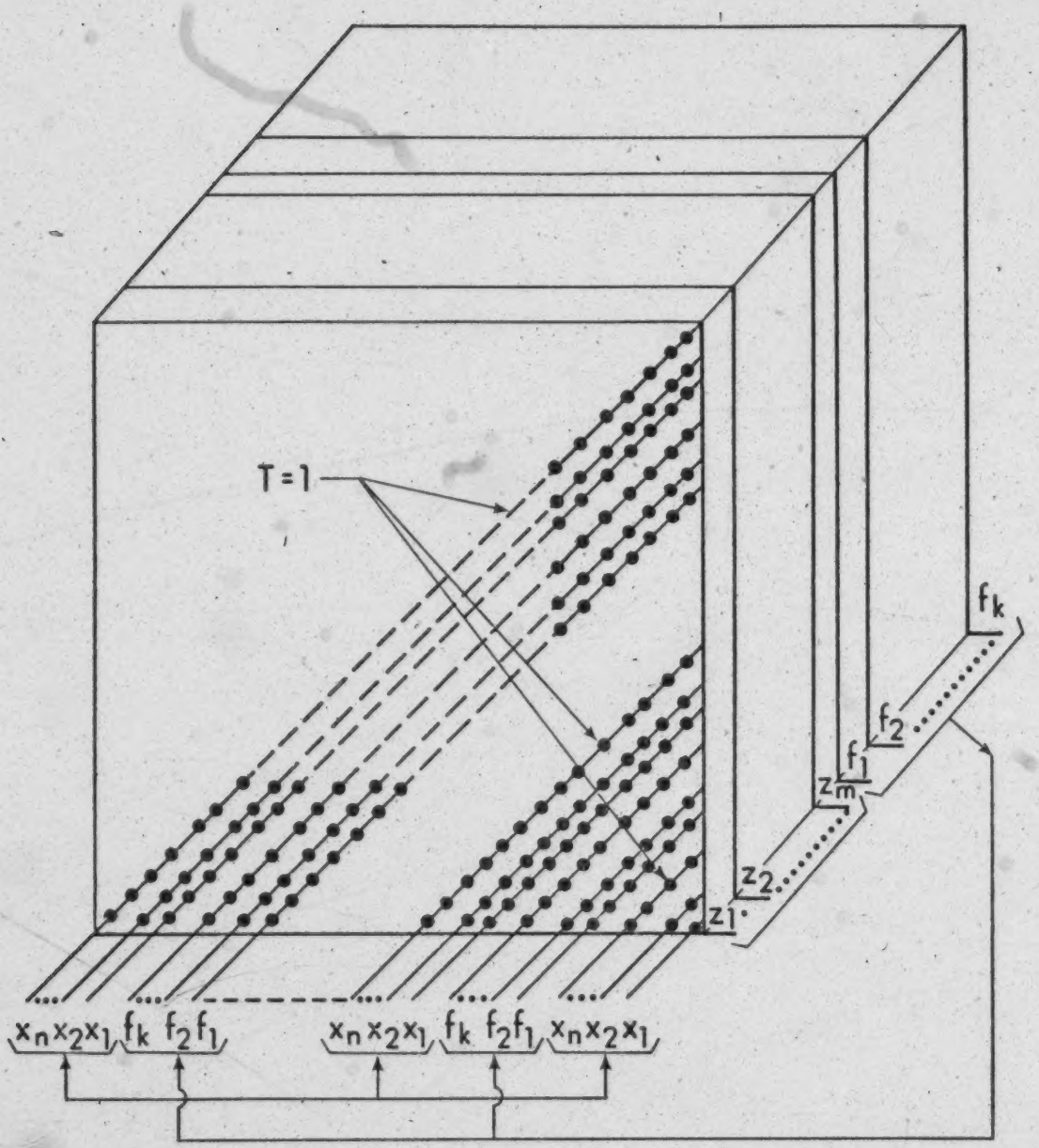


FIG. 33. REALIZATION OF A SYNCHRONOUS SEQUENTIAL MACHINE

CHAPTER 4

SORTING ARRAYS

This chapter presents some sorting arrays suitable for implementation on large-scale-integrated semiconductor technology. One of these arrays is discussed in detail and its performance compared with the one developed by W.H. Kautz [31]. It is shown that, with a slightly more complex cell, the array has a number of advantages over Kautz's array. At the same time, it presents an example in which feedback is used in the combinational circuitry. The most interesting property of the array is that the roles of "smallest" and "largest" are embeded in the same array. Thus, in the array described in this chapter, the largest or the smallest word can be read out from the file. With some circuitry lying external to the main array, additional facilities are incorporated. Some of them are masking, addressing a particular position in the file, shifting the file up or down and using the array as a pushdown or a buffer memory. Finally, some complex-cell sorting arrays using the same concept of feedback are described.

4.1 The Basic Logical Array

First consider the two-dimensional cellular array shown in fig. 34. The array is composed of identical combinational logic cells with three inputs and three outputs. The signal flow is unilateral in the horizontal direction and bilateral in the vertical direction. It is assumed that the inter-cell leads can carry positive as well as negative integral numbers. The combinational circuitry of a cell operates

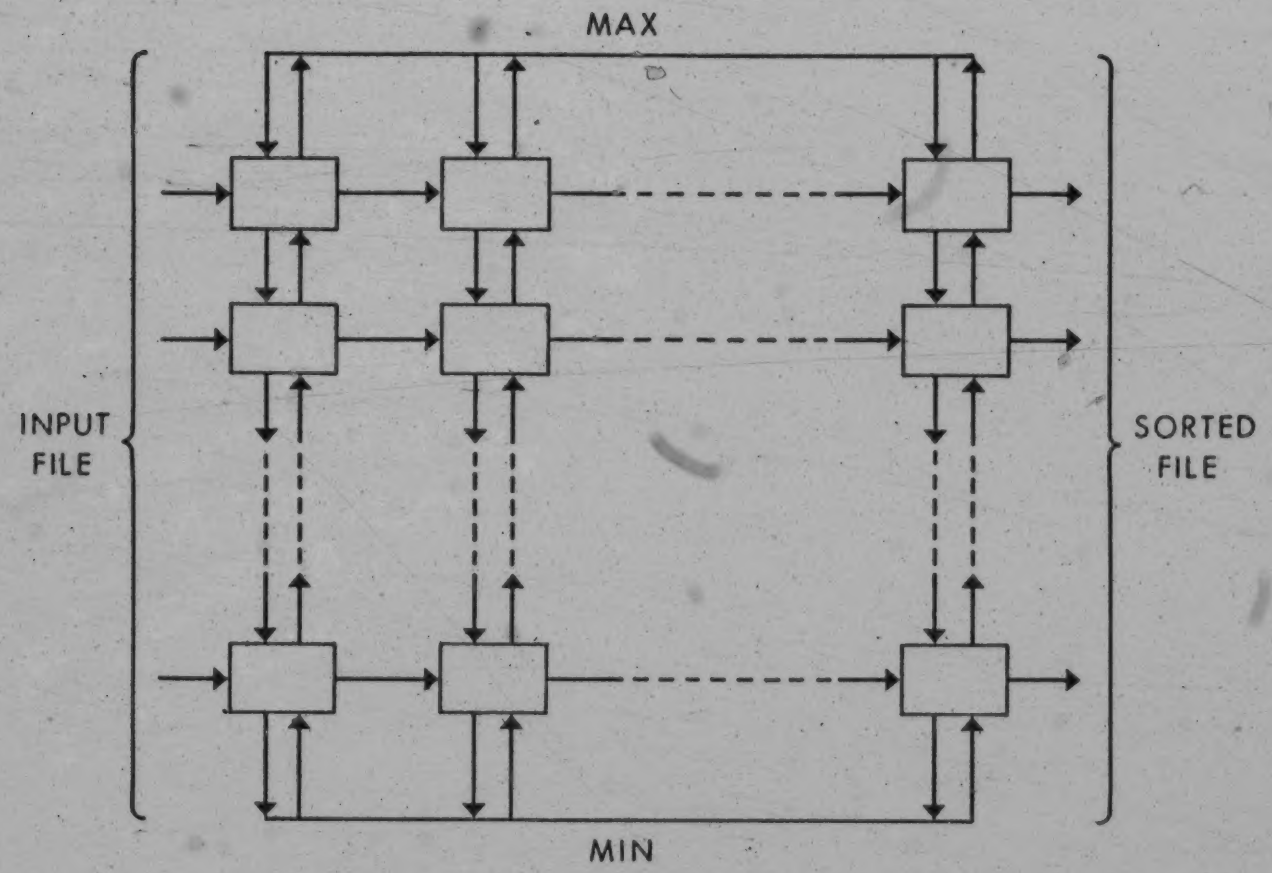


FIG. 34. CELLULAR SORTING ARRAY

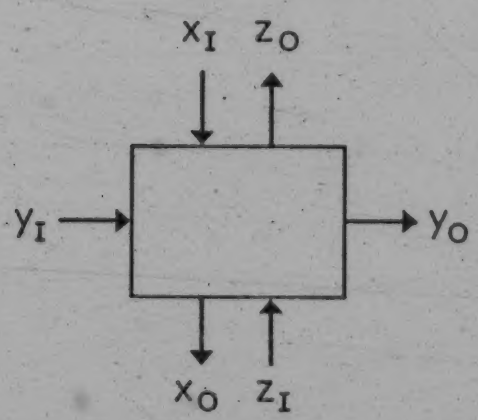


FIG. 35 THE BASIC CELL

under the same number system. The functional capabilities of the cell can be best described by the following logical equations.

$$\left.\begin{aligned} X_0 &= \text{Min}[X_I, Y_I, Z_I] \\ Y_0 &= \text{Mid}[X_I, Y_I, Z_I] \\ Z_0 &= \text{Max}[X_I, Y_I, Z_I] \end{aligned}\right\} \text{------- (2)}$$

Where X_I, Y_I and Z_I are the inputs to the cell, and X_0, Y_0 and Z_0 are the outputs of the cell as shown in fig. 35.

Thus the cell contains comparison circuitry which accepts the three inputs, compares their relative magnitudes and generates the proper outputs. The outputs X_0 and Z_0 represent the smallest and the largest of the three inputs X_I, Y_I and Z_I respectively. The output Y_0 is the middle value.

The boundary inputs to the cells on the top and bottom edges of the array are permanently connected to the largest and the smallest possible numbers (relative to the set of numbers to be sorted) respectively. We will denote these respective numbers by Max and Min. The inputs to the left-most column constitute the set of numbers to be sorted. The numbers are partially sorted at each column of the array. The larger numbers move up while the smaller numbers are forced downward. If the number of columns is sufficiently large, the outputs of the right-most column of the array represent the complete ordering of the inputs entered from the left.

Theorem 1: A column of the cellular array composed of the basic cells can determine the largest and the smallest of a given set of numbers.

Proof: Assume that the left inputs to the column are connected to the given set of numbers. The boundary inputs to the top and bottom cells of the column are Max and Min respectively. First we will show that the signals on inter-cell lines must lie between Max and Min. If possible, assume that a number $S \geq$ Max continues to stay or appear intermittantly on some inter-cell line. Since each output of a cell originates from some input after a small time internal ϵ_t (inertial delay introduced by the cell circuitry), we conclude that there must be a loop in the column along which S flows. This means S is an output of some cell on X line which contradicts eq. (2), since S is greater than the left input to the cell. Similarly, it can be shown that a number $S \leq$ Min can not continue to stay or appear intermittantly on any inter-cell line.

The inputs Max and Min to the boundary cells are returned on Z and X lines respectively. Consider the largest number input to the column at Ith cell from the top. If $I = 1$, this number exits from the right edge of the top cell. If $I > 1$, this number, being the largest input to the cell, is bussed to Z line. It moves up until it reaches the top cell and then exits from the right edge. A similar argument suggests that the smallest input number will move down and exits from the right edge of the bottom cell. Thus the largest and the smallest numbers are obtained from the right edges of the top and bottom cells respectively.

Theorem 2: A set of M numbers can be sorted using at most K columns of the array; where $K = \frac{M}{2}$ if M is even and $K = \frac{M-1}{2}$ if M is odd.

Proof: Let us consider the next largest number input to the same column (Theorem 1) at Jth cell from the top. If $J < I$, this number will exit from the right edge of the same cell. If $J > I$, this number will move up until it reaches the Ith cell. It exits from the right edge of the Ith cell provided that $I > 1$. However, if $I = 1$, this number, being the smallest input to the top cell, is reflected back on X line and then exits from the right edge of the second top cell. Thus the next larger numbers could move up in the same column of the array. Hence the number of columns required to sort a set of numbers depends upon their relative positions at the inputs to the array. Intuitively, if the set of numbers are in the reversed order at the inputs to the array, the number of columns required will be maximum. In such a case, the first column will determine the largest and the smallest numbers of the given set of inputs. These numbers are obtained from the right edges of the top and bottom cells respectively. All other numbers will pass straight through the column. Similarly, the sorting capability of the second column will carry the second largest and the second smallest numbers to the right edges of second top and the second bottom cells respectively. Hence this set of M number will be sorted in K columns of the array; where $K = \frac{M}{2}$ if M is even, and

$K = \frac{M-1}{2}$ if M is odd.

Fig. 36 displays signal flow in such an array for a set of inputs. In the first column, the largest and the smallest numbers are carried to the right edges of the top and the bottom cells respectively. Also the second largest number 9 finds a path up to the right edge of the second bottom cell. All other numbers pass straight through the column. Similarly, the signals in the second column can be determined. The second largest number 9 moves up and exits from the right edge of the second top cell. Consider the signals on intercell lines between the two top cells. The signal on Z line can not go below 9, since there is an input 9 (from bottom) to the second top cell. Also the signal on Z line can not go above 10, since, in that case, the input number 10 (from left) to the top cell will move down and, being the largest input to the second top cell, is reflected back. This forces the signal on Z line, hence in the loop, to return to 10. The outputs on the right edges of the neighbouring cells will change temporarily during this transition, but do stablize as soon as the signal in the loop reaches an equilibrium state (9 or 10). Finally, the signals on intercell lines between the two cells can oscillate between 9 and 10. This case is possible if the signals on X and Z lines are initially 9 and 10 respectively and they exchange their positions after every time interval ε_t. However, it still does not affect the outputs on the right edges of the

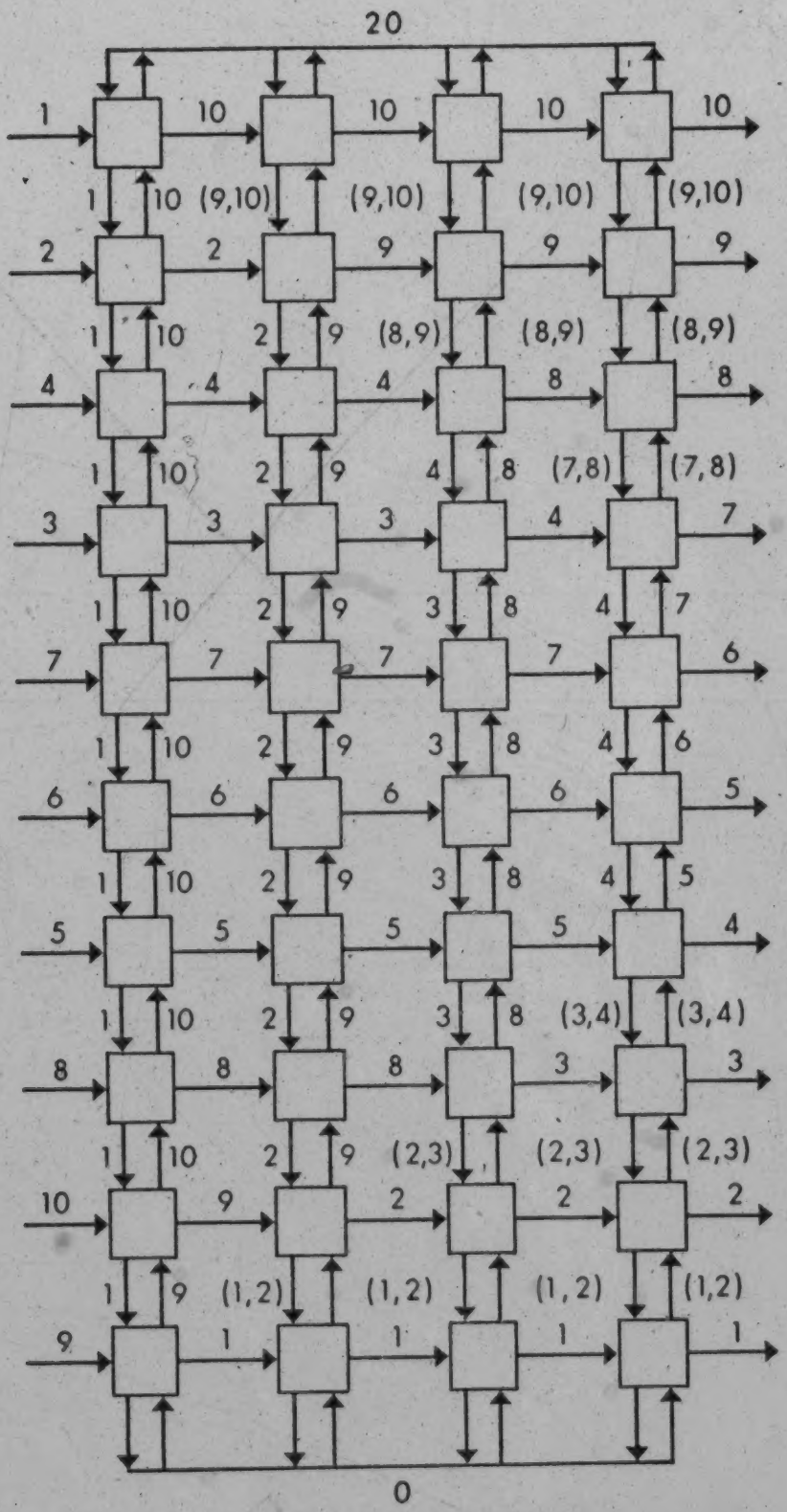


FIG. 36. SIGNAL FLOW IN AN ARRAY

neighbouring cells. The presence of the loop between the two top cells indicates that there is a feedback in the combinational circuitry which leads to oscillations or multiple equilibrium states. Thus, in fig. 36, (X,Y) indicates that the signal on that line can either oscillate between X and Y, or remain constant at any integral level between X and Y. However, these oscillations or multiple equilibrium states do not affect the outputs of the column.

4.2 A Rectangular Sorting Array

We will now describe an array which behaves analogously to the one developed by W.H. Kautz [31]. Fig. 37 illustrates (a) the main array (b) a typical cell and its logic equations. To understand the proper relationship between this array and the basic logical array requires viewing the whole array of Fig. 37 as a single column of Fig. 34.

The numbers are encoded into conventional binary system with the most significant digits at the left ends of the words. Negative numbers are represented by two's complementation. An additional digit 1 or 0 is placed at the left end of the word depending upon whether the number is positive or negative respectively. The number representation for four-digit words is illustrated in Table 5. Note that this particular number representation facilitates comparison of numbers in any row of the array. Sorting with other number representations is possible in the sense described by Kautz. However, those representations require interconnection of subarrays or additional circuitry external to the main array.

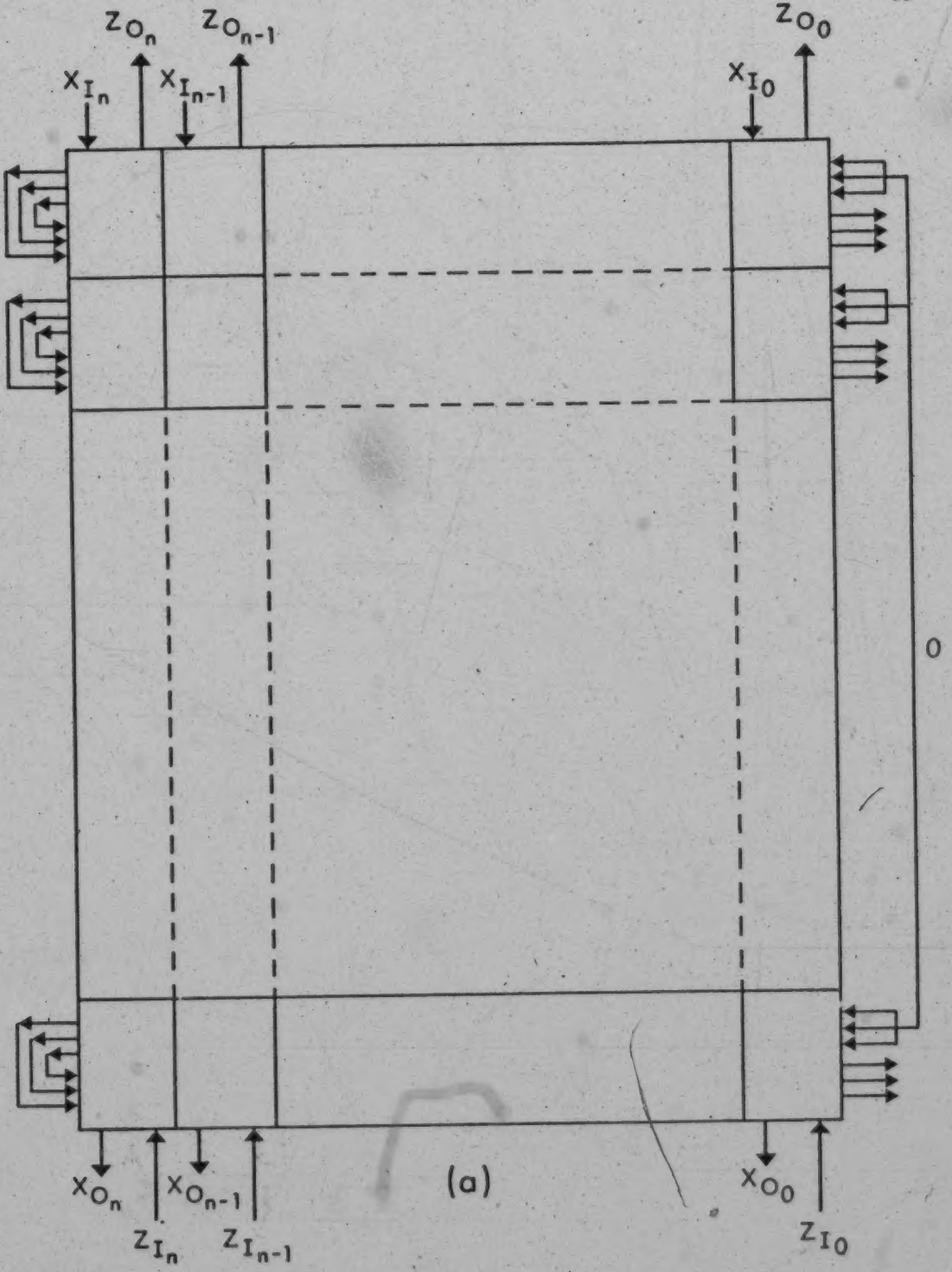


FIG. 37. CELLULAR SORTING ARRAY. (a) THE MAIN ARRAY

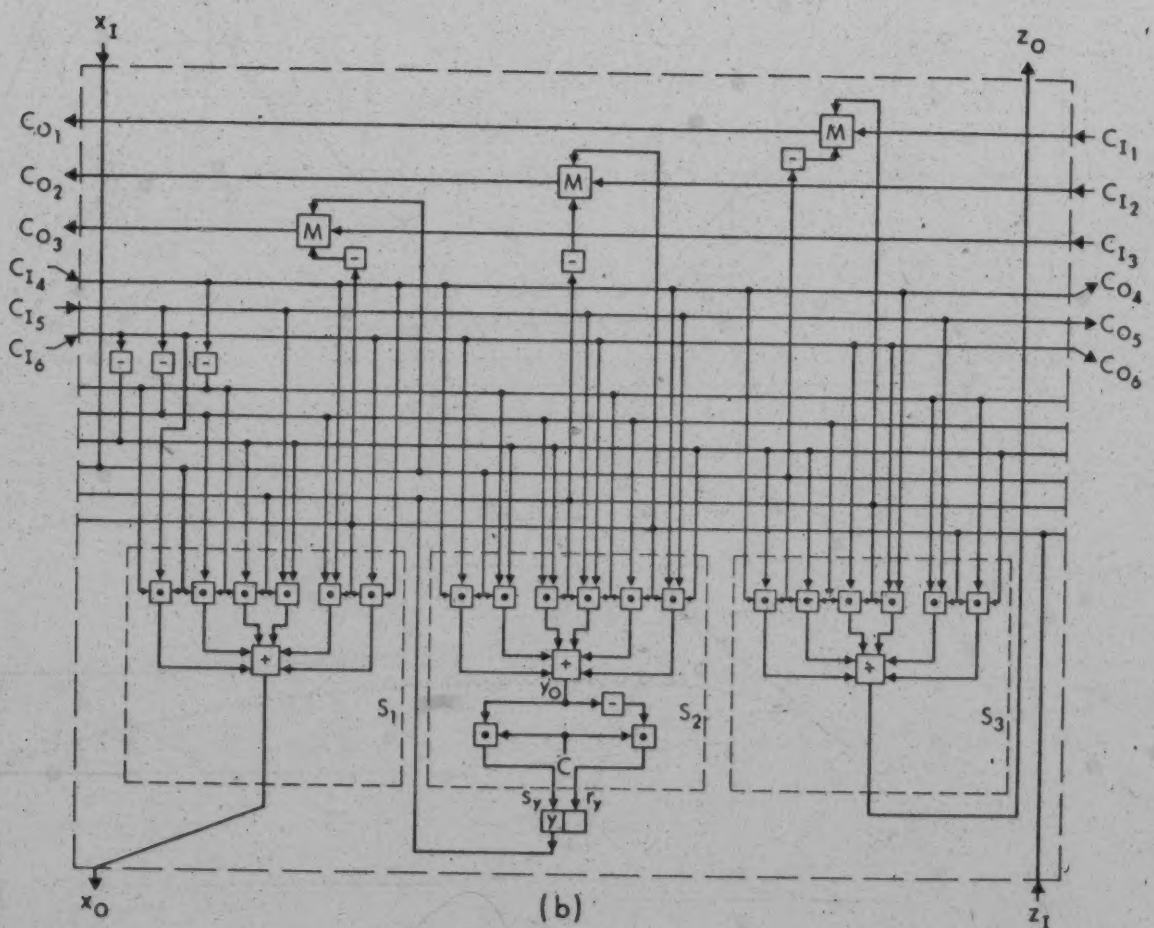


FIG. 37 (CONTINUED). (b) A TYPICAL CELL

Fig. 37(b) (Continued)

The cell logic equations are

$$x_0 = x_I[C_{I_6}+\bar{C}_{I_5}]\bar{C}_{I_4}+y_I[C_{I_5}+\bar{C}_{I_4}]\bar{C}_{I_6}+z_I[C_{I_6}+\bar{C}_{I_5}]C_{I_4}$$

$$y_0 = x_I[C_{I_6}\,C_{I_4}+\bar{C}_{I_6}\,\bar{C}_{I_4}]+y_I[\bar{C}_{I_6}\,\bar{C}_{I_5}+C_{I_6}\,C_{I_5}\,\bar{C}_{I_4}]+z_I[\bar{C}_{I_5}\,\bar{C}_{I_4}+C_{I_5}\,C_{I_4}\,\bar{C}_{I_6}]$$

$$z_0 = x_I[C_{I_4}+\bar{C}_{I_5}]\bar{C}_{I_6}+y_I[\bar{C}_{I_5}+C_{I_4}]C_{I_6}+z_I[C_{I_5}+\bar{C}_{I_6}]\bar{C}_{I_4}$$

$$C_{0_1} = \bar{x}_I y_I+\bar{x}_I C_{I_1}+y_I\,C_{I_1}$$

$$C_{0_2} = \bar{y}_I z_I+\bar{y}_I C_{I_2}+z_I C_{I_2}$$

$$C_{0_3} = \bar{z}_I x_I+\bar{z}_I C_{I_3}+x_I C_{I_3}$$

$$C_{0_4} = C_{I_4}$$

$$C_{0_5} = C_{I_5}$$

$$C_{0_6} = C_{I_6}$$

$$S_y = Cy_0$$

$$r_y = C\bar{y}_0$$

In the above equations $y_I=y(t)$ and $y_0=y(t+1)$, where y(t) represents the state of the flip-flop at time t. C is the clock.

TABLE 5

Number Representation

Number	Binary Representation	Number	Binary Representation
0	1000	-1	0111
1	1001	-2	0110
2	1010	-3	0101
3	1011	-4	0100
4	1100	-5	0011
5	1101	-6	0010
6	1110	-7	0001
7	1111	-8	0000

Each cell of the array contains one flip-flop representing one digit of the word. Thus any row of the array stores one word Y_I with the least significant digit on the right edge of the array. The words X_I and Z_I are represented by the corresponding x_I and z_I lines entering the row. If the words are of maximum length n binary digits, the boundary inputs to the top and bottom rows are $2^{n-1}-1$ and -2^{n-1} respectively. The input terminals on the right hand side of the array are connected to the logical signal 0.

Sorting with this array is performed in the time-domain. The circuitry of each cell consists of three majority gates each associated with one horizontal line. The chain of n majority gates along a row allows the comparison of two numbers, say X_I and Y_I. The carry generated in the left-most cell indicates whether X_I is less than Y_I or not. The words X_I, Y_I and Z_I are compared in pairs. The carries generated in the left-most cell are connected back to the lines (6), (5) and (4) of the same row (external to the array). The signals on these lines should force the largest number upward and the smallest number downward while the digits of the middle number should appear

at the inputs of the corresponding flip-flops of the row. The functional capabilities of the cell required to perform the above task are shown in Table 6.

TABLE 6

Interconnection Code and Output Cell Functions

Interconnection Code			Relative Magnitudes of Numbers	Output Cell Functions		
C_{I_6}	C_{I_5}	C_{I_4}		x_0	y_0	z_0
0	0	0	$X_I \geq Y_I, Y_I \geq Z_I, Z_I \geq X_I$	-	-	-
0	0	1	$X_I \geq Y_I, Y_I \geq Z_I, Z_I < X_I$	z_I	y_I	x_I
0	1	0	$X_I \geq Y_I, Y_I < Z_I, Z_I \geq X_I$	y_I	x_I	z_I
0	1	1	$X_I \geq Y_I, Y_I < Z_I, Z_I < X_I$	y_I	z_I	x_I
1	0	0	$X_I < Y_I, Y_I \geq Z_I, Z_I \geq X_I$	x_I	z_I	y_I
1	0	1	$X_I < Y_I, Y_I \geq Z_I, Z_I < X_I$	z_I	x_I	y_I
1	1	0	$X_I < Y_I, Y_I < Z_I, Z_I \geq X_I$	x_I	y_I	z_I
1	1	1	$X_I < Y_I, Y_I < Z_I, Z_I < X_I$	z_I	x_I	y_I

Note that $y_I = y(t)$ and $y_0 = y(t+1)$, where $y(t)$ represents the state of the flip-flop at time t. From now onward the signals on the feedback lines (6), (5) and (4) of a row will be termed the interconnection code in that row. The interconnection code 000 ($X_I = Y_I = Z_I$) leads to optional entries in the output functions of the cell. The case $X_I < Y_I, Y_I < Z_I, Z_I < X_I$ is not possible. The reason for particular entries in the output functions corresponding to the interconnection code 111 will be discussed later.

The logical equations for a cell can be derived from Table 6 and are shown in fig. 37. Assuming that the combinational circuitry of a cell follows these equations, then on occurrence of the clock pulse, the digits appearing at the inputs are stored in the corresponding flip-flops of the row. The new file is a partial ordering of the old file. Thus the file of M words can be sorted in approximately M/2 sorting cycles.

We will now discuss the performance of this array and its advantages over Kautz's array. We will assume that the file is already loaded but not necessarily sorted.

1) To read out the words in order of their size, largest to smallest, feed the min (minimum possible word 00---0) from the top and carry out the sorting cycle repeatedly. During each cycle, the next larger word will be read out from the top while the min is stored somewhere in the file. The read out cycle also allows partial ordering of the words if the file is previously not sorted. If the words are desired in opposite order, a similar process can be carried out from the bottom. The word fed from the bottom will be of maximum size (11----1).
2) Write-read can be carried out simultaneously. In this case, the number to be inserted in the file is fed from the top and the sorting cycle executed. The word is written somewhere in the file while the largest word is read out from the top.
3) The largest and the smallest numbers can be read out simultaneously while two numbers are being inserted in the file.
4) A search for all numbers $Y \geq X_1$ can be conducted in a simple manner. The number X_1 is fed from the top and the sorting cycle is executed repeatedly until the word X_1 is returned. The word Y read out

during each cycle satisfies the relation $Y \geq X_1$. Similarly, the words $Y \geq X_2$ can be read out from the bottom.

5) Unlike Kautz's array, our array does not require external registers and their additional programming. Consider the registerless operation in Kautz's array. First, it is not even possible to clear the file. This is due to the fact that if Min is fed from the top, it passes through all the rows and exits from the bottom. The file of words remains unchanged during each sorting cycle. Second, if the largest word is desired, a single 1 must be placed on Z line of the top row before the array is pulsed. But this is not possible in the registerless operation of the array. Third, if the words in the file are read out in order, smallest to largest, by entering repeatedly Max from the top, the array is left full of 1's at the end. Now it is not possible either to clear the file or to enter a new word in the file. These observations indicate a distinct disadvantage in the registerless operation of Kautz's array over our array.

5) Sorting over a set of digits (called the key position) is straight forward. The array is augmented by vertical lines bussed along the columns of the array. These lines are fed from the outputs of a mask register. The complete array with mask register, the basic cell and its logical equations are shown in fig. 38. The comparison is inhibited in all the columns whenever m=0. The key position is first selected by appropriately setting the mask register. The array is then cycled approximately M/2 times to sort the file of M words to the key position. If the key position is changed, resorting can be accomplished by executing the sorting cycle repeatedly. Unlike Kautz's array, no left-most digits (tag digits) are reserved to indicate the re-sorting

status. Furthermore, no additional circuitry for the left-most column and programming of the register etc. is required.

7) The array may also be employed for various other purposes having nothing to do with sorting. A possible arrangement to incorporate some additional facilities is shown in fig. 39. A bidirectional address register W, a gating network and two flip-flops P_1 and P_2 are used external to the array. The logical equations for a typical cell in the gating network are also given in the same figure. The flip-flops P_1 and P_2 define four modes shown in Table 7.

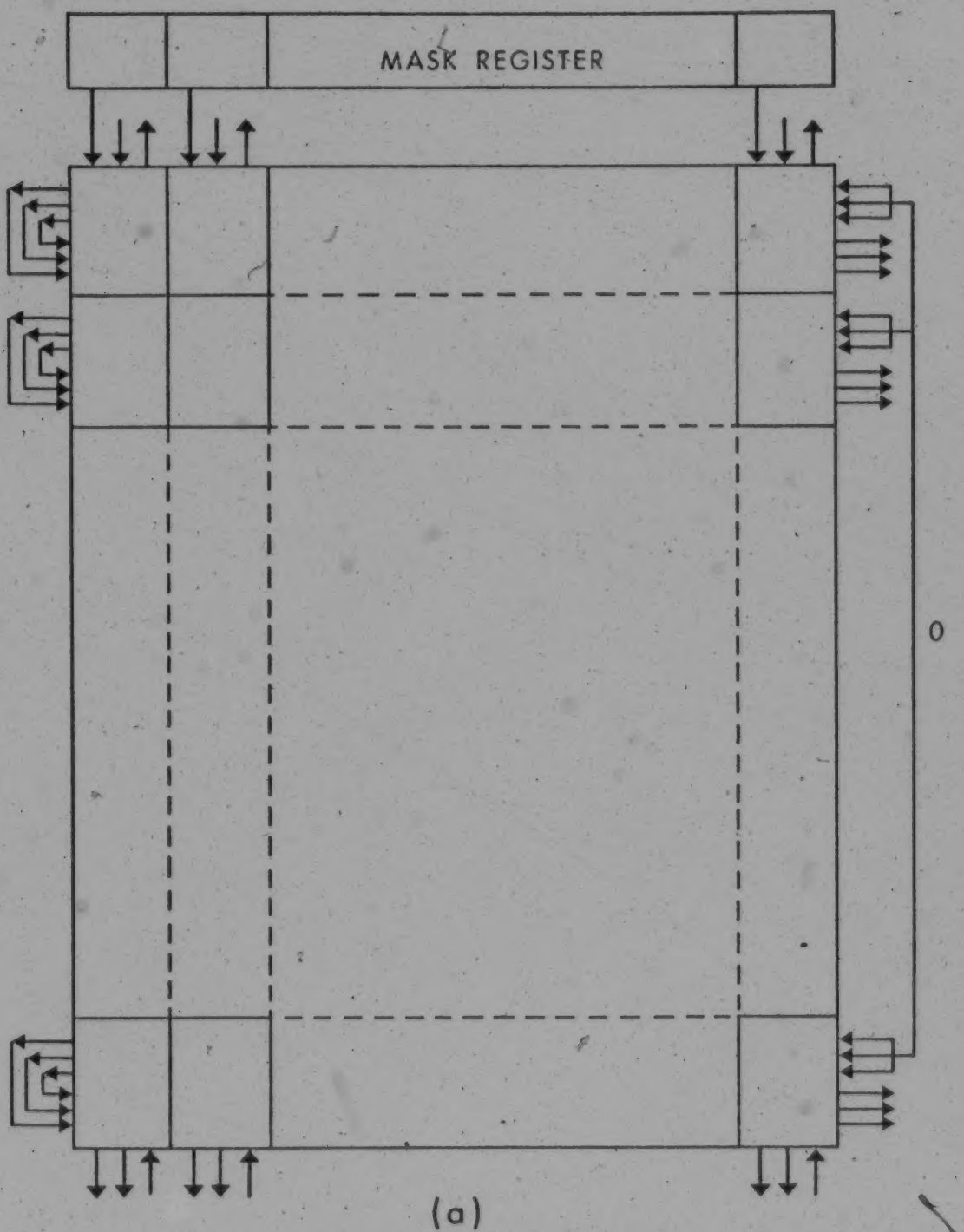


FIG. 38. CELLULAR SORTING ARRAY WITH MASK REGISTER
(a) THE MAIN ARRAY

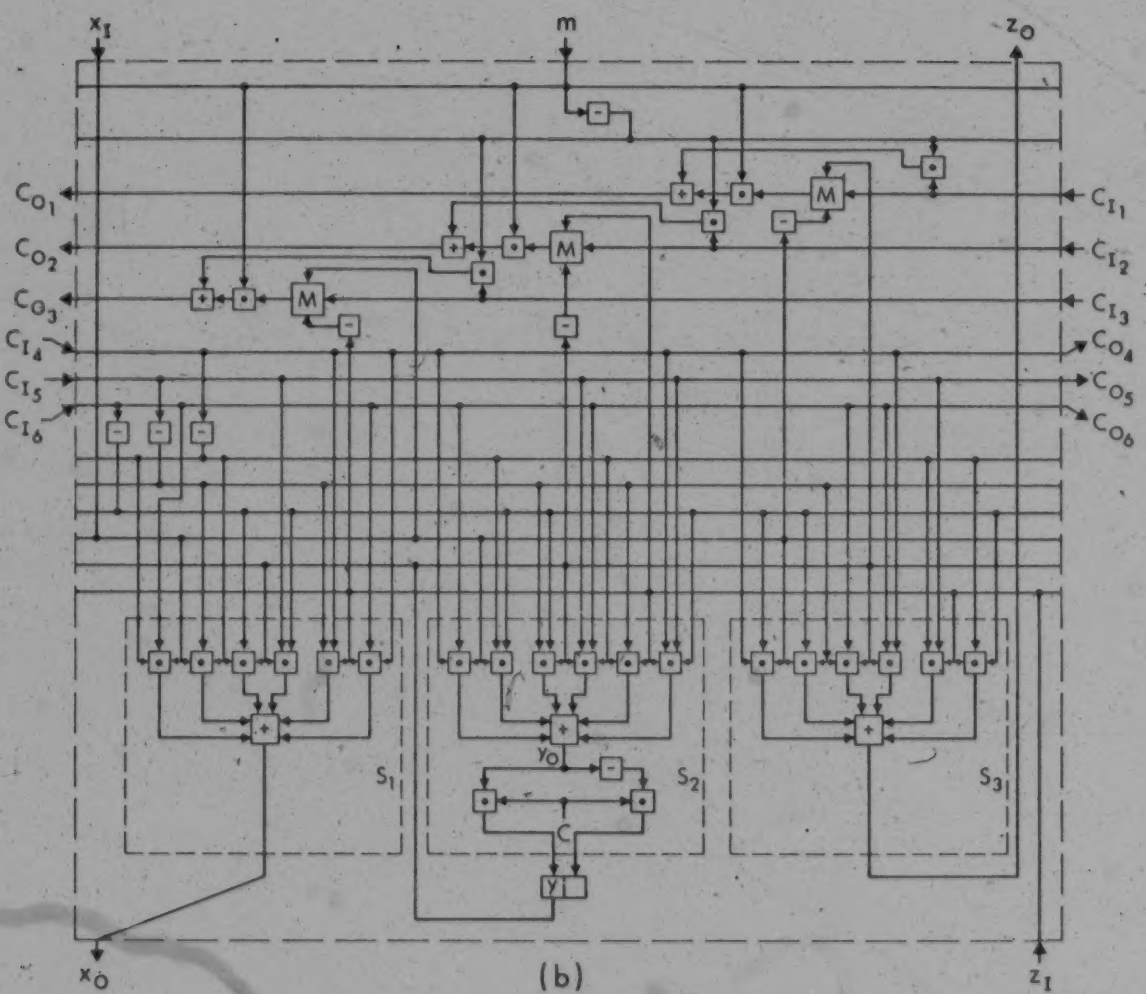


FIG. 38 (CONTINUED). (b) A TYPICAL CELL

Fig. 38(b) (Continued).

The cell logic equations are

$$x_0 = x_I[C_{I_6}+\bar{C}_{I_5}]\bar{C}_{I_4}+y_I[C_{I_5}+\bar{C}_{I_4}]\bar{C}_{I_6}+z_I[C_{I_6}+\bar{C}_{I_5}]C_{I_4}$$

$$y_0 = x_I[C_{I_6}C_{I_4}+\bar{C}_{I_6}\bar{C}_{I_4}]+y_I[\bar{C}_{I_6}\bar{C}_{I_5}+C_{I_6}C_{I_5}\bar{C}_{I_4}]+z_I[\bar{C}_{I_5}\bar{C}_{I_4}+C_{I_5}C_{I_4}\bar{C}_{I_6}]$$

$$z_0 = x_I[C_{I_4}+\bar{C}_{I_5}]\bar{C}_{I_6}+y_I[\bar{C}_{I_5}+C_{I_4}]C_{I_6}+z_I[C_{I_5}+\bar{C}_{I_6}]\bar{C}_{I_4}$$

$$C_{O_1} = m[\bar{x}_I y_I+\bar{x}_I C_{I_1}+y_I C_{I_1}]+\bar{m}C_{I_1}$$

$$C_{O_2} = m[\bar{y}_I z_I+\bar{y}_I C_{I_2}+z_I C_{I_2}]+\bar{m}C_{I_2}$$

$$C_{O_3} = m[\bar{z}_I x_I+\bar{z}_I C_{I_3}+x_I C_{I_3}]+\bar{m}C_{I_3}$$

$$C_{O_4} = C_{I_4}$$

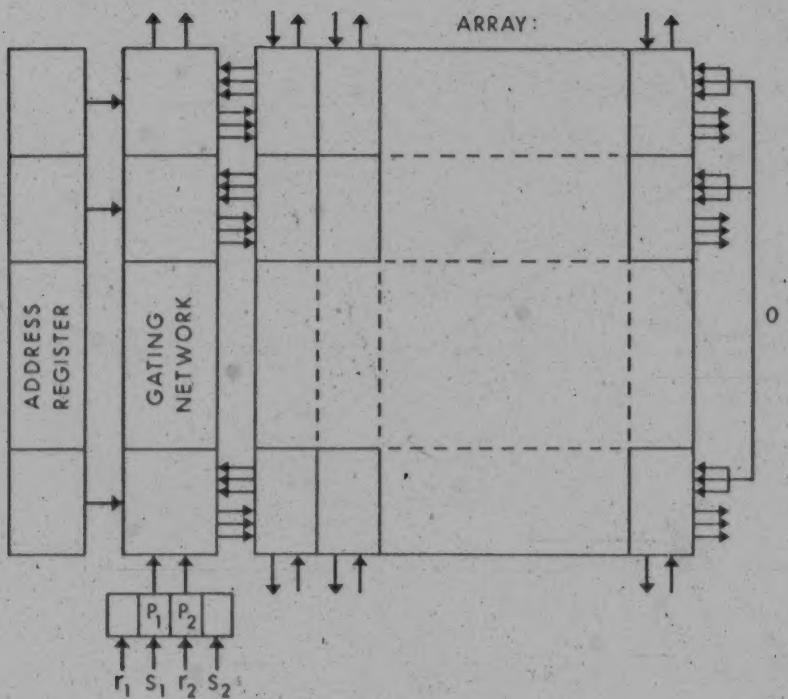
$$C_{O_5} = C_{I_5}$$

$$C_{O_6} = C_{I_6}$$

$$S_y = Cy_0$$

$$r_y = C\bar{y}_0$$

In the above equations $y_I=y(t)$ and $Y_0=y(t+1)$, where $y(t)$ represents the state of the flip-flop at time t. C is the clock.



TYPICAL CELL IN THE GATING NETWORK:

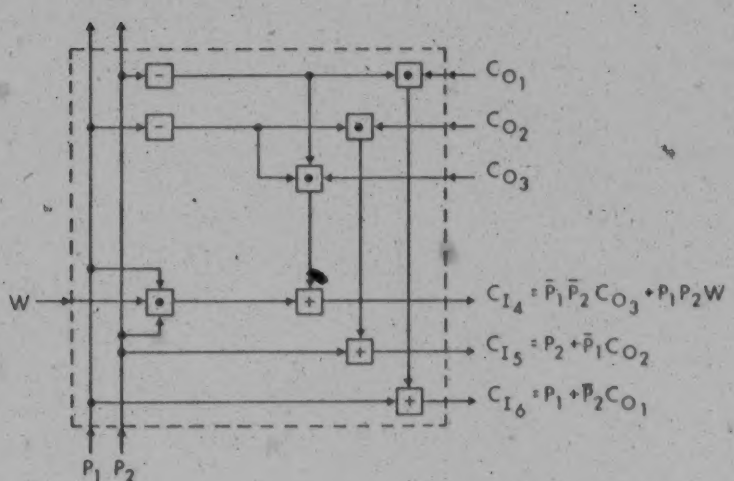


FIG. 39. SORTING ARRAY AND ADDRESSED MEMORY.

Table 7

Four Modes in the Rectangular Sorting Array

State of the Flip-Flops		Mode	Interconnection Code in any Row		
P_1	P_2		C_{I_6}	C_{I_5}	C_{I_4}
0	0	Sorting	C_{O_1}	C_{O_2}	C_{O_3}
0	1	Push Down	0	1	0
1	0	Push Up	1	0	0
1	1	Addressing	1	1	W

When $P_1=P_2=0$, the array behaves normally as already discussed. The lines (6), (5) and (4) of any row are connected through the gating network to the respective lines (1), (2) and (3) of the same row. This particular mode is termed the sorting mode (Table 7).

When $P_1=P_2=1$, the lines (6) and (5) of all the rows are connected to the logical signal 1. The line (4) of any row is connected to the output of the corresponding flip-flop in the address register W. Thus, the interconnection code in any row can be 110 or 111. We will later discuss how this mode can be used to address a particular position in the file.

When $P_1=0$ and $P_2=1$, the interconnection code in any row is 010. In this mode the file has the capability of shifting downward.

When $P_1=1$ and $P_2=0$, the interconnection code in any row is 100. If the array is pulsed in this mode, the complete file is pushed up

through one position.

A. Addressed Memory

If the flip-flops P_1 and P_2 are set to 1, the array is set to the addressing mode. Any position in the file can now be addressed by appropriately setting the address register W. The interconnection code in any row is 110 or 111 depending upon the output of the corresponding flip-flop in the address register W. The interconnection code 110 in a row allows the transfer of a word from top to bottom and bottom to top. If the interconnection code is 111, the number fed from the top appears at the input of the corresponding register while the word already stored in the register is bussed to the z-lines. This is the reason for particular entries in the output functions for the interconnection code 111 (Table 6). Thus the interconnection code 111 in a row represents the addressed position. To insert a word in a particular position in the file, the corresponding flip-flop in the register W is set to 1 and the other flip-flops are set to 0. The word is fed from the top. It passes through all the rows and appears at the addressed position in the file. The word already stored there is bussed to the z-lines. It passes through all the rows and, finally, exits from the top edge of the array. On occurrence of the clock pulse, the word is inserted in the addressed position while the number already stored there is read out from the top edge of the array. In this way, a number of words can be inserted to the addressed positions. If desired, the file can then be sorted by executing the sorting cycle repeatedly.

B. Pushdown Memory

The array may be employed as a pushdown memory (last in - first out). To enter the word, set P_1=0 and P_2=1, and pulse the array. The complete file is moved down through one position and the new word fed from the top edge is inserted in the first row. To read out the word, set P_1=1 and P_2=0, and pulse the array. The complete file is pushed up through one position. The word last fed in is read out from the top edge of the array.

C. Buffer Memory

To use the array as a buffer memory (first in - first out), a single 1 is placed in the address register W (initially in row 1) to address the row where the next word should be inserted. To enter the words, the array is set to the addressing mode by setting P_1=P_2=1. The address register W is shifted one row downward after each entry. To read out the next word, set P_1=1, P_2=0 and pulse the array. The desired word is read out from the top edge. During each read out cycle, the address register W is shifted up through one position.

4.3 Fault-Testing

A major problem faced by logical designers of LSI circuitry is that of fault testing. In this section, we will first describe a procedure to test the array assuming no failures and then the consequences of failures [25]. For simplicity, we will make some assumptions. First, all failures which occur are constant with time. Second, only one cell in any column of the array is faulty. Third, the feedback

lines can be disconnected and the interconnection code in any row can be controlled separately. Finally, no buses or interconnections have failed. In the following procedure, the input words X_I and Z_I are assumed to have values Max (11----1) and Min (00----0). It can be easily shown that this reduces the number of tests required to detect and locate the fault.

Step 1: Set the interconnection code 110 in all the rows. This allows transfer of words from top to bottom and bottom to top in any row. Successively apply the four possible combinations of input words X_I and Z_I and observe the outputs X_0 and Z_0. The outputs X_0 and Z_0 correspond correctly to the inputs X_I and Z_I respectively if all the cells are functioning properly for this interconnection code.

Step 2: To test the i-th row, change its interconnection code to 111. Set X_I=Min and pulse the array. This inserts Min in the i-th row.

Step 3: Set a particular interconnection code for the i-th row. Vary the input words X_I and Z_I, and observe the outputs X_0 and Z_0, and the carries generated in the left-most cell of the i-th row. Repeat the test for other interconnection codes.

Step 4: Change the interconnection code of i-th row to 111. Set X_I=Max and pulse the array. This inserts Max in the i-th row. Repeat Step 3.

Step 5: Repeat steps 2, 3 and 4 as above, one row at a time until the complete array is tested.

In the absence of failures, the observed outputs will correspond exactly to the predetermined correct outputs. If we observe an incorrect output from a cell, we conclude that either the cell is faulty or it has received an incorrect input. For example, in Step 1, assume that the output word X_0 does not correspond correctly to the input word X_I at J-th bit. To locate the faulty cell, first change the interconnection code of the first row to 001, vary the input words X_I and Z_I, and observe the output words X_0 and Z_0. The conclusion can be drawn from Table 8.

TABLE 8

Fault-detection in the Rectangular Sorting Array

X_0	Z_0	Conclusion	Possible Faults
(a) Correct	Correct	Cell (1,J) is faulty	Switch S_1
(b) Correct	Incorrect	Cell (1,J) is faulty	Switch S_1 and Switch S_3
(c) Incorrect	Correct	Any cell (X,J), x=1, 2----ℓ can be faulty. ℓ is the number of rows in the array.	-----
(d) Incorrect	Incorrect	Cell (1,J) is faulty	Switch S_1 and Switch S_3

If case (c) of Table 8 is observed, change the interconnection code of 2nd row to 001 and that of first row to 110. Once again, vary the input words X_I and Z_I, and observe the outputs X_0 and Z_0. The conclusion now follows from Table 9.

TABLE 9

Fault-detection in the Rectangular Sorting Array

	X_0	Z_0	Conclusions	Possible Faults
(a)	Correct	Correct	Cell (1,J) is perfect but cell (2,J) is faulty	Switch S_1
(b)	Correct	Incorrect	Cell (1,J) or Cell (2,J) is faulty	Switch S_1 in cell (1,J) or Switch S_1 and Switch S_3 in Cell (2,J)
(c)	Incorrect	Correct	Any cell (x,J), x=2, 3----ℓ can be faulty	-----
(d)	Incorrect	Incorrect	Cell (2,J) is faulty	Switch S_1 and Switch S_3

If case (c) of Table 9 is observed, the test procedure continues for subsequent rows. If case (b) is observed, it is possible to carry out an alternative procedure to locate which of the two cells is faulty. First change the interconnection code of the second row to 011. Set Z_I=Min and pulse the array. This inserts Min in the second row. Now set X_I=Min and observe the carries generated from the left-most cell of the second row. If all carries are 0's, then the top input to the J-th cell of the second row is not stuck at 1. To insert Max in the second row, set Z_I=Max and pulse the array. Now change X_I=Max and observe the carries generated from the left-most cell of the second row. If all carries are 0's, then the top input to the J-th cell of the second row is not stuck at 0. If the carries generated in both the cases are correct, we must conclude that switch S_1 in cell (1,J) is functioning properly.

A similar procedure can be developed to locate the fault, if detected, while carrying out step 3 or 4. To locate the faulty majority gate, the input words X_I and Z_I must be varied from Max to Min continuously.

4.4 A Cubic Sorting Array

One of the simplest version of the basic array is a three-dimensional cellular array in which sorting can be accomplished in the space-domain. Fig. 40 illustrates (a) one plane of the array and (b) a typical cell and its logical equations. The inputs to the first plane represent the file of words to be sorted while the inputs to the top and bottom edges of all planes are permanently connected to Max (11----1) and Min (00----0). The words are partially sorted at each plane. The partially ordered file of words penetrates through the plane and falls on the next plane. The ordered file of words is received from the other side of the cubic array. Similar to the Theorems(1) and (2), it is easier to show that one plane can determine the largest and the smallest of the given file, and K planes can sort a file of M numbers; where K=M/2 if M is even and $K=\frac{M-1}{2}$ if M is odd.

Note that the network of the cubic array consists of only combinational circuitry. We will now show how the cubic array or a plane

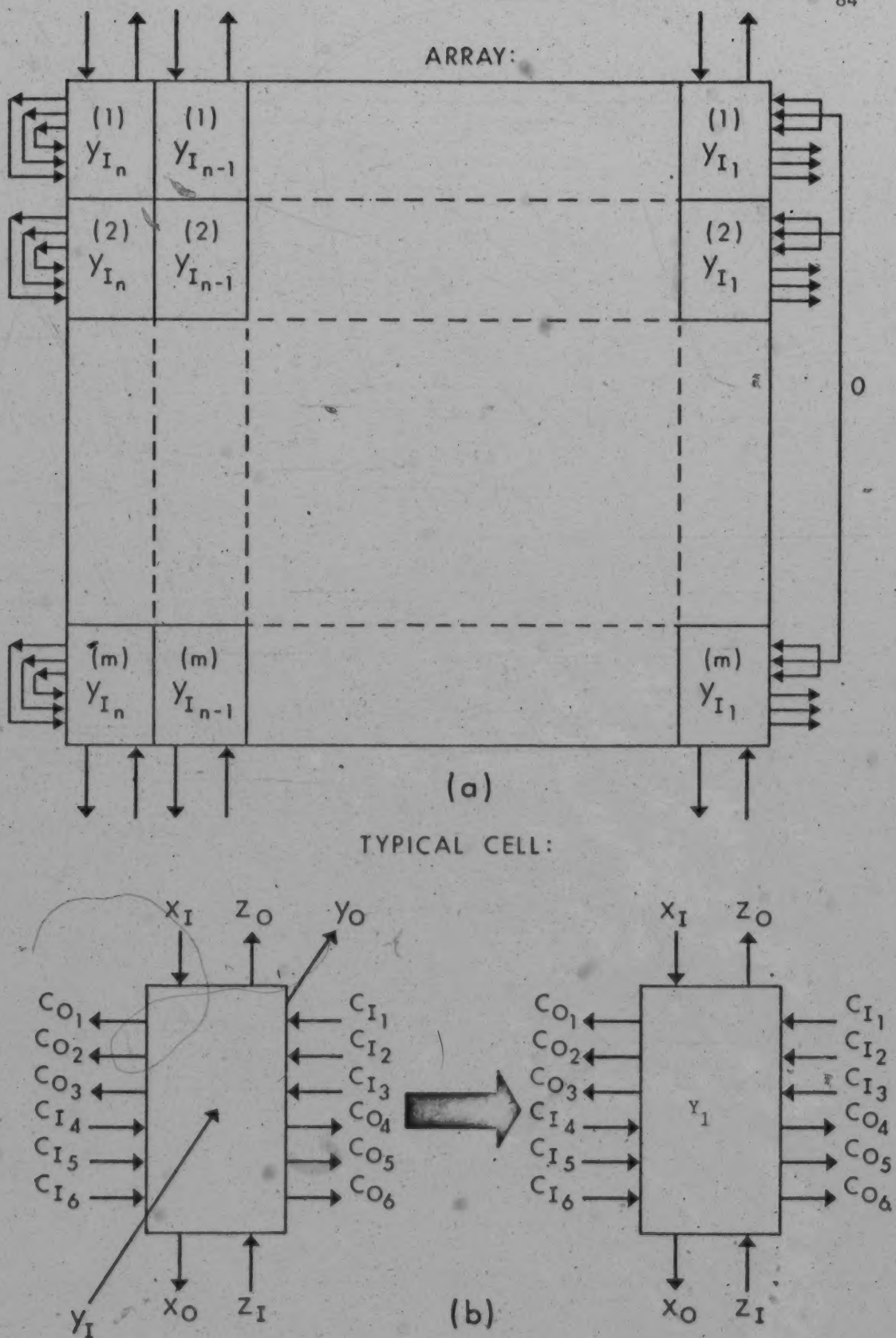


FIG. 40. CELLULAR ARRAY FOR SORTING IN THE SPACE DOMAIN

The cell logic equations are

$$x_0 = x_I[C_{I_6} + \bar{C}_{I_5}]\bar{C}_{I_4} + y_I[C_{I_5} + \bar{C}_{I_4}]\bar{C}_{I_6} + z_I[C_{I_6} + \bar{C}_{I_5}]C_{I_4}$$

$$= x_I[C_{I_6}\ C_{I_4} + \bar{C}_{I_6}\ \bar{C}_{I_4}] + y_I[\bar{C}_{I_6}\ \bar{C}_{I_5} + C_{I_6}\ C_{I_5}\ \bar{C}_{I_4}] + z_I[\bar{C}_{I_5}\ \bar{C}_{I_4} + C_{I_5}\ C_{I_4}\ \bar{C}_{I_6}]$$

$$z_0 = x_I[C_{I_4} + \bar{C}_{I_5}]\bar{C}_{I_6} + y_I[\bar{C}_{I_5} + C_{I_4}]C_{I_6} + z_I[C_{I_5} + \bar{C}_{I_6}]\bar{C}_{I_4}$$

$$C_{0_1} = \bar{x}_I y_I + \bar{x}_I C_{I_1} + y_I\ C_{I_1}$$

$$C_{0_2} = \bar{y}_I z_I + \bar{y}_I C_{I_2} + z_I C_{I_2}$$

$$C_{0_3} = \bar{z}_I x_I + \bar{z}_I C_{I_3} + x_I C_{I_3}$$

$$C_{0_4} = C_{I_4}$$

$$C_{0_5} = C_{I_5}$$

$$C_{0_6} = C_{I_6}$$

of the array can be used to realize a set of combinational switching functions. A possible arrangement for realizing p functions of K variables is shown in Fig. 41. The left K-inputs in any row are the K literals $\dot{x}_1, \dot{x}_2$ ---- $\dot{X}_K$ representing the min-term $\dot{x}_1 \dot{x}_2$ ---- $\dot{x}_K, \dot{x}_1$ denotes x_1 or $\bar{x}_1$. The right p inputs in the same row are the binary digits representing the outputs of p functions corresponding to this min-term. Thus the array will have dimensions $(P+K)x2^K$. The boundary inputs to the top and bottom edges of the array is the minimum possible number (00----0). For any input combination of K variables, the first K inputs of some row will be all 1. The word in that row will then be the largest in the file. The sorting capabilities of the array will carry this number to the top edge. The right p bits of this number represent the outputs for p functions corresponding to the present input combination of variables. Thus to realize p functions f_1, f_2----$, f_p$, we need to find the vertices contained in each function. The input to the (I,K+J)th cell is 1 if and only if the function f_J contains the vertex $\dot{x}_1 \dot{x}_2$----$\dot{x}_K$, where

$$I = 1 + \sum_{l=1}^{K} C_l \, 2^{K-l}$$

and $$C_l = \begin{cases} 1, \dot{x}_l = x_l \\ 0, \dot{x}_l = \bar{x}_l \end{cases}$$

The output leads for the functions f_1, f_2----$, f_p$ are shown in Fig. 41.

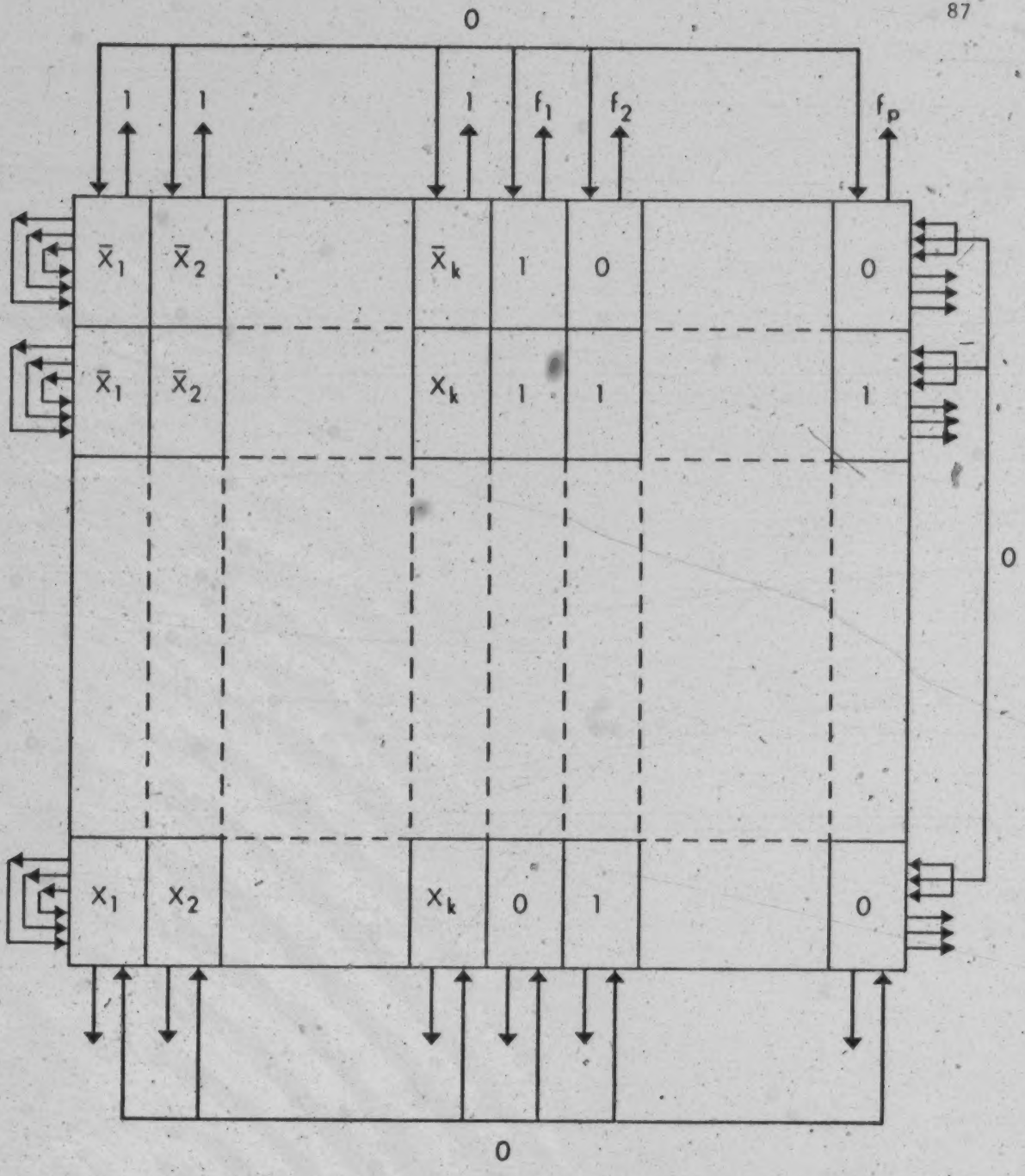


FIG. 41. REALIZATION OF A SET OF COMBINATIONAL SWITCHING FUNCTIONS

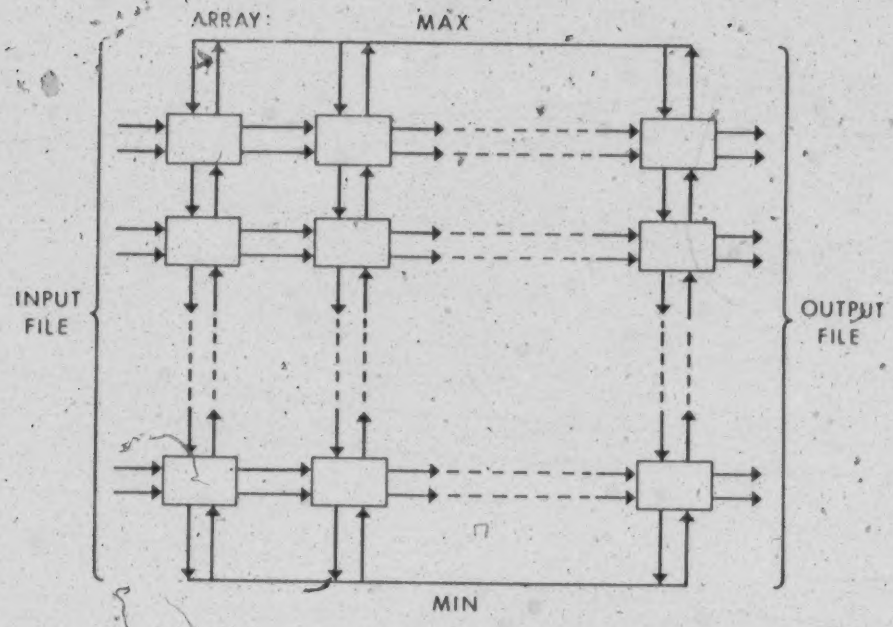
4.5 Cell Complexity Versus the Array Size

Some complex-cell sorting arrays can be developed on the same concept of feedback. The total array size for sorting a set of numbers can be reduced by increasing the complexity of the cell and interconnection structure. We will illustrate this phenomenon with the help of some examples.

Fig. 42 illustrates a sorting array, its basic cell and logical equations. Once again the individual cells of the array are identical and contain only combinational logic. Note that there are four numbers which are compared in each cell and this can be interpreted as the complexity of the cell. An upper bound to the number of cells needed to sort a file of M numbers is $\lceil \frac{M}{2} \rceil \times \lceil \frac{M}{2} \rceil$. $\lceil n \rceil$ denotes the smallest integer larger than or equal to n. Fig. 43 displays signal flow in such an array.

A rectangular sorting array analogus to the one developed in section 11 can be easily derived. Each row in the array will store two words. The equi-weighted digits of the words will be represented by two flip-flops in a cell. The comparison of four numbers in pairs will require six horizontal lines running from right to left. The six bits of the interconnection code will be carried by additional six horizontal lines. Interesting properties of the array can be devised by ingenious designers.

Instead of increasing the horizontal lines, one could add vertical lines and develop a number of different sorting arrays on the same concept. Fig. 44 displays a sorting array, its basic cell and



TYPICAL CELL:

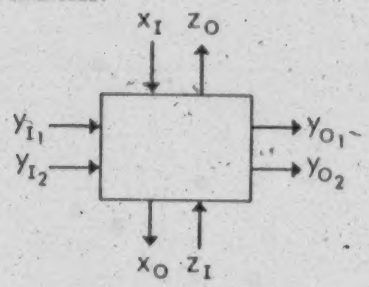


FIG. 42. CELLULAR SORTING ARRAY. THE CELL LOGIC EQUATIONS ARE

$$Z_o(t) = \text{Max}_1[Y_{I_1}(t), Y_{I_2}(t), X_I(t), Z_I(t)]$$

$$Y_{o_1}(t) = \text{Max}_2[Y_{I_1}(t), Y_{I_2}(t), X_I(t), Z_I(t)]$$

$$Y_{o_2}(t) = \text{Min}_2[Y_{I_1}(t), Y_{I_2}(t), X_I(t), Z_I(t)]$$

$$X_o(t) = \text{Min}_1[Y_{I_1}(t), Y_{I_2}(t), X_I(t), Z_I(t)] \quad Z_o(t) \geq Y_{o_1}(t) \geq Y_{o_2}(t) \geq X_o(t)$$

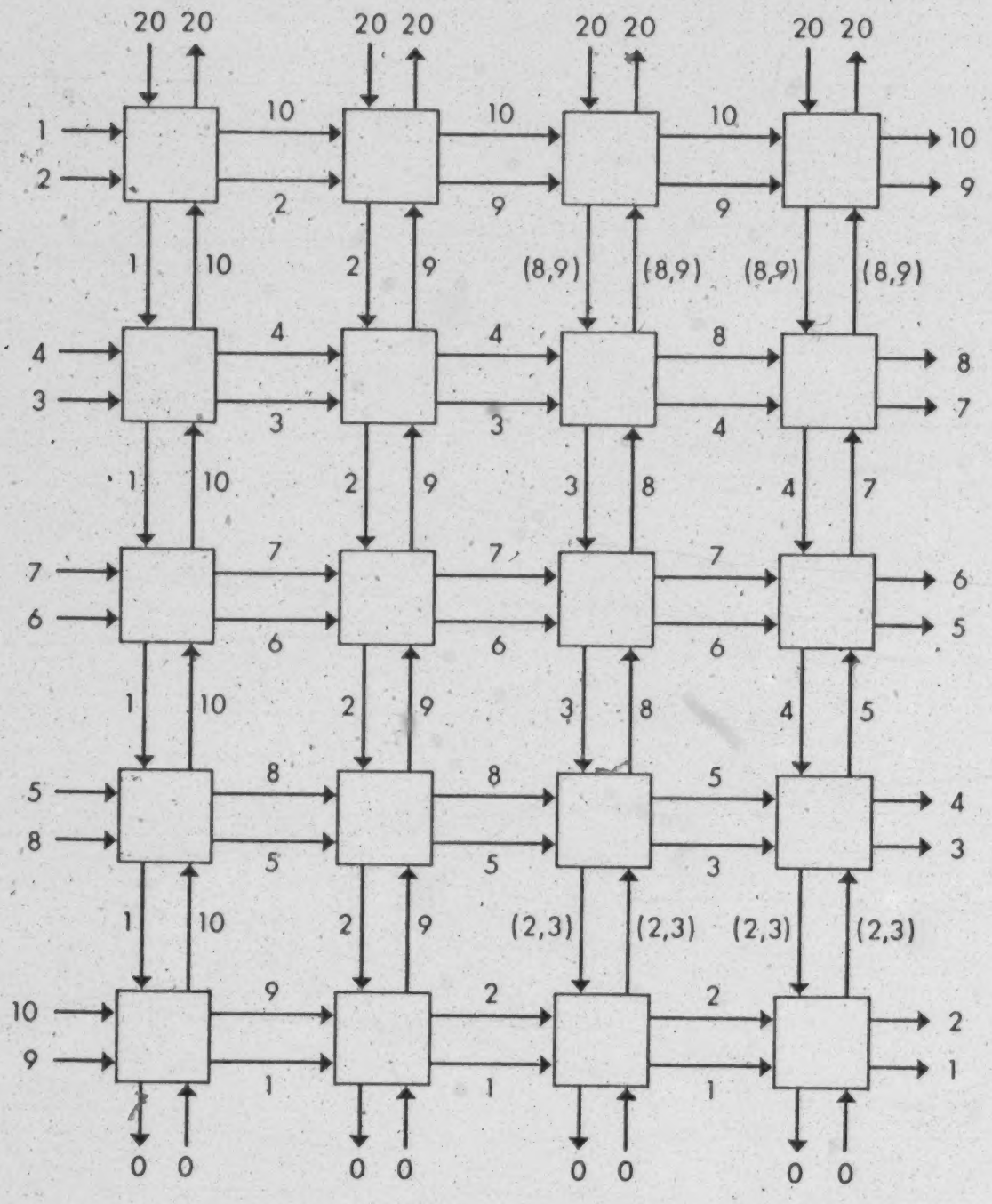
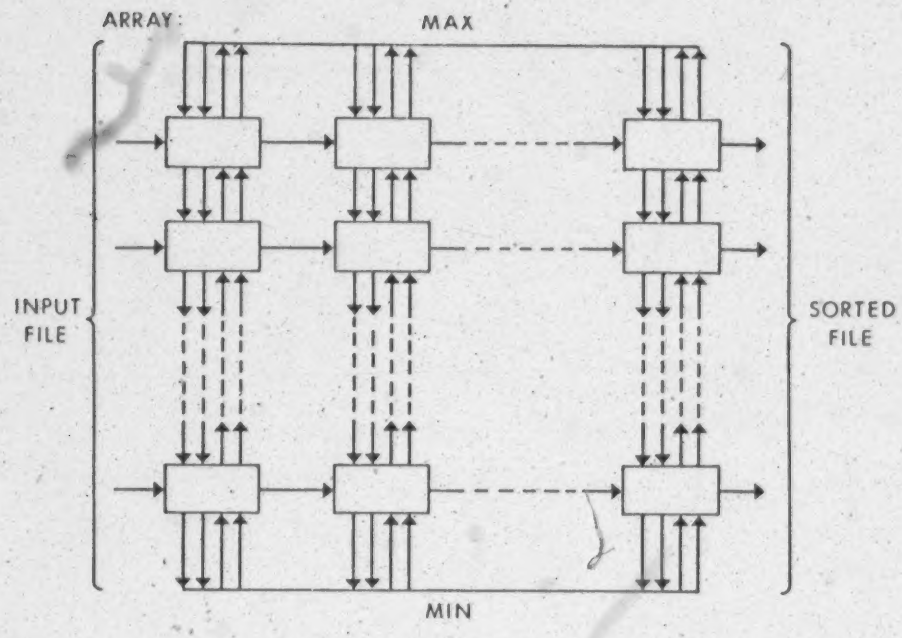


FIG. 43. SIGNAL FLOW IN AN ARRAY



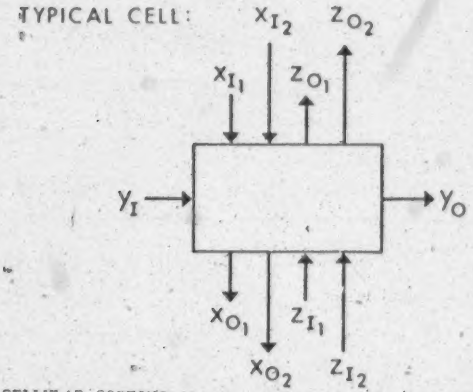


FIG. 44. CELLULAR SORTING ARRAY. THE CELL LOGIC EQUATIONS ARE:

$$Z_{O_1}(t) = \text{Max}_1[X_{I_1}(t), X_{I_2}(t), Y_I(t), Z_{I_1}(t), Z_{I_2}(t)]$$

$$Z_{O_2}(t) = \text{Max}_2[X_{I_1}(t), X_{I_2}(t), Y_I(t), Z_{I_1}(t), Z_{I_2}(t)]$$

$$Y_O(t) = \text{Mid}[X_{I_1}(t), X_{I_2}(t), Y_I(t), Z_{I_1}(t), Z_{I_2}(t)]$$

$$X_{O_2}(t) = \text{Min}_2[X_{I_1}(t), X_{I_2}(t), Y_I(t), Z_{I_1}(t), Z_{I_2}(t)]$$

$$X_{O_1}(t) = \text{Min}_1[X_{I_1}(t), X_{I_2}(t), Y_I(t), Z_{I_1}(t), Z_{I_2}(t)]$$

the cell logic equations. While five numbers are compared in each cell, the number of cells needed to sort a file of M words is at most M x ⌈M/4⌉. The array can be easily converted to a rectangular sorting array in which sorting can be accomplished in the time-domain.

In general, if there are n horizontal lines and p vertical lines, then an upper bound to the number of cells needed to sort a file of M words is $\lceil \frac{M}{n} \rceil \times \lceil \frac{M}{p} \rceil$.

CHAPTER 5

CONCLUSIONS

In this thesis we have presented some cellular arrays for the synthesis of combinational switching functions and synchronous sequential machines. The properties of the array are comparable to those of some other cellular structures. The edge fed arrays seem to be the simplest and most versatile structures and attractive for implementation on LSI. The edge fed universal logic array for four variables developed in the second chapter is really very efficient. It requires only eight cells (the additional cell is only used as a buffer cell) and, more-over, the number of inputs to the array approaches to that of Muller and Preparata's ULM [28]. The cubic array can be used to realize a set of combinational switching functions. It leads to higher packing density in comparison to that of other arrays. With respect to the realization of synchronous sequential machines, the array enjoys a simple unique cell structure and uniform interconnections. Finally, the array can be used as an interconnection network or a crossbar switch [29]. The sorting array presented in the last chapter could find application in any computing system in which sorting capability is desired.

In future research on combinational and sequential cellular structures, it will be important to design universal logic arrays which optimize a set of design parameters. Some of these parameters have already been listed in Chapter 2. The research so far has failed to produce efficient arrays. Intutively, it appears that there exists

some sort of trade-off between the growth rate and the number of cut-point inputs per cell. Consideration should be given in future research to develop cellular arrays that have more inputs per cell. Such arrays, with suitable synthesis algoritham, can be expected to be much more efficient than the present ones. With the advent of microminature monolithic integrated circuits, it appears that in future generations of microelectronic devices each cell will be fabricated on an extremely small area of the substrate. Thus it will be very costly to implement cutpoint operation for each cell. In future research, it is deemed desirable to develop efficient edge fed universal logic arrays.

In further research, techniques should be developed for the synthesis of a set of combinational switching functions on a cellular array. Such an array with unit delays (external to the array) can be used to realize an arbitrary synchronous sequential machine. Minimization of the size of these arrays based on proper state assignment is a problem worthy of investigation.

In this thesis we were mainly concerned with combinational and sequential cellular structures. However, cellular arrays can be designed for various other functions such as arithmatic, error correcting, pulse generating, counting and analog circuitry.

REFERENCES

[1] R.C. Minnick, "Survey of Microcellular Research," J. ACM, Vol. 14, pp. 203-241, April 1967.

[2] R.C. Minnick, "Cutpoint Cellular Logic," IEEETEC, Vol. EC-13, pp. 685-698, Dec. 1964.

[3] Amar Mukhopadhyay and Harold S. Stone, "Cellular Logic," in Recent Developments in Switching Theory, ed. A. Mukhapadhyay, Academic Press, pp. 255-313, 1971.

[4] J. Sklansky, "General synthesis of tributary switching networks," IEEE Trans. Electron. Comput. EC-12, pp. 464-469, 1963.

[5] S.Y. Levy and R.O. Winder, "A note on tributary switching network," IEEE Trans. Electron. Comput. EC-13, pp. 148-151, 1964.

[6] C.D. Weiss, "The Characterization and properties of Cascade Realizable Switching Functions," IEEE Trans. Comput. C-18, pp. 624-633, 1969.

[7] H.S. Stone, "On the number of equivalence classes of functions realizable by cellular cascades," In Proc. Nat'l. Symp. on the Impact of Batch Fabrication on Future Computers, pp. 81-87, 1965.

[8] J. Sklansky, A. Korenjak and H.S. Stone, "Canonical tributary networks. IEEE Trans. Electron. Comput. EC-14, pp. 961-963, 1964.

[9] R.A. Short, "Two-rail Cellular Arrays," In AFIPS Conf. Proc. Proc. 27, pt. 1, Washington, D.C., Spartan, pp. 355-369, 1965.

[10] Bernard Elspas, "The Theory of Multirail Cascades," in Recent Developments in Switching Theory, ed. A. Mukhopadhyay, Academic Press, pp. 315-367, 1971.

[11] L.M. Spandorfer and J.V. Murphy, "Synthesis of Logic Functions on an array of integrated circuits. Scientific Rep. No. 1 for UNIVAC project 4645, AFCRL 63528, Contract AF 19(628) 2907, Sperry Rand Corp., UNIVAC Engineering Centre, Bluebell, Pa., Oct. 31, 1963.

[12] R.A. Short, "Cellular Linear-input Logic," Final Report, SRI project 4122, Contract AF19(628)-498, STanford Research Institute, Menlo Park, Calif., AFCRL 64-6; DDC No. AD 433802, Feb. 1964.

[13] R.H. Canaday, "Two-dimensional iterative logic. MIT Electronic System Lab., Rep. ESL-R-21o, MIT Project DSR 9891, Contract AF-33(657)-11311, AF Task 4421 (Sept. 1964).

[14] R.H. Canaday, "Two-dimensional iterative logic," Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Pt. 1, pp. 343-353.

[15] S. Amarel, G. Cooke and R.O. Winder, "Majority Gate Network," Scientific Rep. to AFCRL, No. 793, AD 268907 (Aug. 1961).

[16] R.C. Minnick and R.A. Short, "Cellular Linear-input Logic," Final Report, SRI Project 4122, Contract AF 19(628)-498, Stanford Research Institute, Menlo Park, Calif., AFCRL 64-6; DDC No. AD 433802, Feb. 1964.

[17] K.K. Maitra, "Cascaded Switching Networks of two-input flexible cells," IRE Trans. EC-11, 2 (April 1962), pp. 136-143.

[18] R.C. Minnick, "Cobweb Cellular Arrays," Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, pt. 1, pp. 327-341.

[19] M.M. Newborn, "A Synthesis Technique for Binary Input Binary Output Synchronous Sequential Machines," IEEETC, Vol. C-17, pp. 697-699, July 1968.

[20] T.F. Arnold, C.J. Tan, and M.M. Newborn, "Iteratively Realized Sequential Circuits," IEEETC, Vol. C-19, pp. 54-66, Jan. 1970.

[21] M.M. Newborn and T.F. Arnold, "Universal Module for Bounded Signal Fan Out Synchronous Sequential Circuits," IEEETC, Vol. C-21, pp. 63-79, Jan. 1972.

[22] D. Ferrari and A. Grasselli, "A Cellular structure for Sequential Switching Circuits," IEEETC, Vol. EC-15, pp. 354-367, June 1966.

[23] A.D. Friedman, "Feedback in Synchronous Sequential Switching Circuits," IEEETC, Vol. EC-15, pp. 354-367, June 1966.

[24] J.C. Huang, "A Universal Cellular Array," IEEETC, Vol. C-20, pp. 317-320, March 1971.

[25] K.J. Thurber, "Fault Location in Cellular Arrays," in 1969 Proc. Fall Joint Comput. Conf., AFIPS Conf. Proc., Vol. 35, Montvale N.V.: AFIPS Press, 1969, pp. 81-88.

[26] W.H. Kautz, "Fault testing and diagnosis in combinational digital circuits," IEEE Trans. Comput., Vol C-17, April 1968, pp. 352-366.

[27] S.B. Akers, "A rectangular Logic Array," IEEETC, Vol. C-21, pp. 848-857, Aug. 1972.

[28] Harold S. Stone, "Universal Logic Modules," in Recent Developments in Switching Theory, ed. A. Mukhopadhyay, Academic Press, pp. 229-253, 1971.

[29] W.H. Kautz, "Programmable Cellular Logic," in Recent Developments in Switching Theory, ed. A. Mukhopadhyay, Academic Press, pp. 369-422, 1971.

[30] S. N. Kukreja and I. Ngo Chen, "Combinational and Sequential Cellular Structures," IEEE Trans. Computers, Vol. C-22, pp. 813-823, Sept. 1973.

[31] W.H. Kautz, "Cellular logic in-memory arrays," IEEE Trans. Computers, Vol. C-18, pp. 719-727, August 1969.

[32] K.E. Batcher, "Sorting networks and their applications," 1968 Spring Joint Computer Conf., AFIPS Proc., Vol. 32, Washington, D.C.: Thompson, 1968, pp. 307-314.

[33] S.S. Yau and M. Orric, "Fault diagnosis and repair of cutpoint cellular arrays," IEEE Trans. Comput., Vol. C-19, Mar. 1970, pp. 259-262.